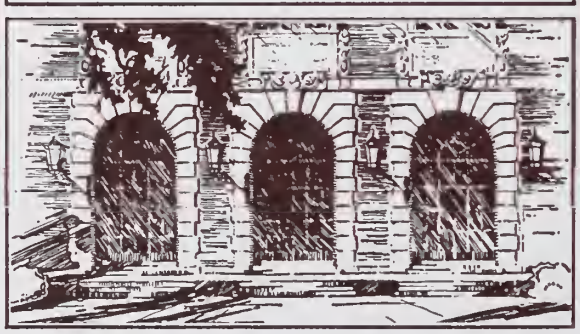
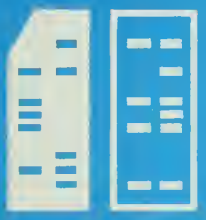


Report No. UIUCDCS-R-72-370-4

SOUPAC PROGRAM DESCRIPTIONS

STATISTICALLY ORIENTED USERS PROGRAMMING AND CONSULTING

February 1, 1972



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. UIUCDCS-R-72-370-4

SOUPAC PROGRAM DESCRIPTIONS

STATISTICALLY ORIENTED USERS PROGRAMMING AND CONSULTING

February 1, 1972

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This manual has been prepared by the Statistical Consultants of the Department of Computer Science, University of Illinois at Urbana-Champaign, as documentation for the system of statistical programs known as SOUPAC. SOUPAC has been written by the statistical consultants of this department in an effort to provide a broad range of standard statistical procedures which are of use to the academic community at large. Inquiries about SOUPAC should be directed to 138 Digital Computer Laboratory.

Previous editions

370	December 1, 1969
370-1	January 20, 1970
370-2	September 15, 1970
370-3	March 1, 1971

TABLE OF CONTENTS

VII. Factor Analysis Package

BINORMAMIN
CENTROID FACTOR ANALYSIS
COMMUNALITY ESTIMATION
ITERATIVE FACTOR ANALYSIS
JACOBI
OBLIMAX ROTATION
ORTHOGONAL PROCRUSTES; KAISER'S TECHNIQUE FOR RELATING FACTORS BASED ON DIFFERENT SAMPLES
SCATTER PLOTS
PRINCIPAL AXIS FACTOR ANALYSIS (Eigenvalues and Vectors)
PROCRUSTES (Oblique Case)
SQUARE ROOT FACTOR ANALYSIS
THREE-MODE FACTOR ANALYSIS
UNRESTRICTED MAXIMUM LIKELIHOOD FACTOR ANALYSIS
VARIMAX FACTOR ROTATION
VARISIM ROTATION
TUCKER-MESSICK POINTS OF VIEW ANALYSIS

VIII. Econometrics Package

ECONOMETRIC REDUCED FORM AND RESIDUAL ANALYSIS
K-CLASS ESTIMATION
LINEAR PROGRAMMING
QUADRATIC PROGRAMMING
EXACT RESTRICTED LEAST SQUARES
STOCHASTIC RESTRICTED LEAST SQUARES
THREE STAGE LEAST SQUARES ESTIMATION
THE TRANSPORTATION PROBLEM

IX. Spectral Analysis Section

AUTOCORRELATIONS
CROSPA

X. Scale Analysis Package

CLIQUE ANALYSIS
PAIRED COMPARISONS
SCALOGRAM ANALYSIS

XI. Probit Analysis Section

PROBIT

CROSS REFERENCE

Statistical Technique or Estimate	SOUPAC Source
Alpha factor analysis	ITERATIVE
Analysis of Variance	BALANOVA, T-TEST
Analysis of Covariance	T-TEST, UMAVAC*
Analysis of Covariance Structures	RESTRICTED
Autocorrelations	AUTOCORRELATIONS
Autocovariances	AUTOCORRELATIONS, CROSPA
Bartlett's Chi-Square Test	UMAVAC*
Beta Coefficients	CANONICAL, CORRELATIONS, MULTIPLE STEP-WISE
Biserial Correlations	BISERIAL
Boudon Method for Path Analysis	PATH
Canonical Correlations	CANONICAL
Canonical Factor Analysis	ITERATIVE
Centroid Factors	CENTROID
CHOLESKY Decomposition	MATRIX
Chi-Square	CANONICAL, CHI-SQUARE, FREQUENCY, PROBIT
Clique analysis	CLIQUE
Coherence	CROSPA
Communality estimation	COMMUNALITIES
Correlations	AUTOCORRELATIONS, BISERIAL, CANONICAL, CORRELATIONS, DISCRIMINANT, MISSING, MULTIPLE, OBLIMAX, PARTIAL, STEP-WISE, K-CLASS, EXACT, STOCHASTIC
Covariance, Analysis of (See Analysis of Covariance)	
Covariance Matrix	CORRELATION, K-CLASS, MULTIPLE, STEP-WISE THREE STAGE, EXACT, STOCHASTIC
Covariance Matrix of Coefficients	K-CLASS

Statistical Technique or Estimate	SOUPAC Source
Covariance Matrix of Reduced Form Residuals	ECONOMETRIC
Covariance Matrix for Residuals	ECONOMETRIC, EXACT
Cross-products Matrix	CORRELATIONS, DISCRIMINANT, K-CLASS, MULTIPLE, EXACT, STOCHASTIC
Cross-Spectrum	CROSPA
Cross-Tabulation	FREQUENCY
Cross-variances	CROSPA
Data Processing	MATRIX, STANDARD, TRANSFORMATIONS
Determinent of a Matrix	MATRIX
Determination, Coefficient of	MULTIPLE, STEPWISE, K-CLASS
Deviation Covariance Matrix	MULTIPLE
Deviation (partial) Correlation matrix	MULTIPLE
Diagonal Method Factor Analysis	SQUARE
Dichotomous Data	BISERIAL, SCALOGRAM
Discriminant Function	DISCRIMINANT
Discriminant Scores	DISCRIMINANT
Dispersion Matrix	DISCRIMINANT
Distribution Analysis	FIT, FREQUENCY, KOLMOGOROV
Duncan Method for Path Analysis	PATH
Durbin-Watson Coefficients	ECONOMETRIC, MULTIPLE
Dwyer Extension Analysis	PROCRUSTES
Eckart-Young Decomposition	POINTS OF VIEW
Eigenvalues	DISCRIMINANT, JACOBI, K-CLASS PRINCIPAL AXIS, THREE-MODE
Eigenvectors	JACOBI, PRINCIPAL AXIS, THREE-MODE
Exact Restricted Least Squares	EXACT
F-Ratios	BALANOVA, DISCRIMINANT, MULTIPLE, STEP-WISE, T-TEST, EXACT, STOCHASTIC
Factor Analysis	CENTROID, ITERATIVE, JACOBI, POINTS OF VIEW, PRINCIPAL AXIS, SQUARE ROOT, THREE-MODE, UNRESTRICTED, ORTHOGONAL PROCRUSTES, PROCRUSTES

Statistical Technique or Estimate	SOUPAC Source
Factor comparisons	FACTOR ANALYSIS PREFACE, KAISER FACTOR MATCHING
Factor Scores (See Factor Analysis)	
Filtering	CROSPA
Frequency Distributions	FREQUENCY
Gamma Coefficients	FREQUENCY
Goodman and Kruskal Coefficients	FREQUENCY
Hotelling's T	UMAVAC*
Individual differences	POINTS OF VIEW
Integer Programming	INTEGER*
Iterative Principal Axis Solution	ITERATIVE
K-Class Estimation	K-CLASS
Kolmogorov-Smirnov Statistic	KOLMOGOROV
Kronecker Product	MATRIX
Lagged Variables	MATRIX
Mann-Whitney U	MANN-WHITNEY
Lambda Coefficients	CANONICAL, DISCRIMINANT, FREQUENCY
Latin Squares	UMAVAC*
Least Squares Factor Fitting	PROCRUSTES
Limited Information Estimation	K-CLASS
Linear Programming	LINEAR
MANOVA (Multivariate Analysis of Variance)	UMAVAC*
Matrix Operations	MATRIX
Maximum Likelihood Factor Analysis	UNRESTRICTED
Means	CORRELATIONS, DISCRIMINANT K-CLASS, MISSING, MULTIPLE, T-TEST, STANDARD, EXACT, STOCHASTIC
Minkowski Space	TORSCA*
Missing Data Correlations	MISSING
Moving Averages	STANDARD

Statistical Technique or Estimate	SOUPAC Source
Multiple correlation	CANONICAL, MULTIPLE, STEP-WISE
Multiple-dimensional Scaling	TORSCA*
Noncard, non SOUPAC generated input	MATRIX (INPUT operation)
Nonmetric Multiple-dimensional scaling	TORSCA*
N-way Chi-Squared	FREQUENCY
Oblique Factor Rotation	BINORMAMIN, OBLIMAX, PROCRUSTES
Orthogonal Factor Rotation	VARIMAX, VARISIM, ORTHOGONAL PROCRUSTES
Orthogonal Simple Structure	VARIMAX, VARISIM, ORTHOGONAL PROCRUSTES
Paired Comparisons	PAIRED
Partial Correlation Coefficients	REGRESSION
Phi Coefficients	use CORRELATIONS
Plots	SCATTER PLOTS
Point Biserial Correlations	CORRELATIONS
Polynomial Fits	MINROS*; TPOLY*
Predicted Values	ECONOMETRIC, MULTIPLE
Primary Factor Pattern	BINORMAMIN, OBLIMAX, PROCRUSTES
Principal Components (See Factor Analysis)	
Prior Information	EXACT, STOCHASTIC
Probit Analysis	PROBIT, MULTIVARIATE
Punched Card Output	MATRIX, Appendix B
Quadratic Programming	QUADRATIC, INTEGER*
Quadrature Spectrum	CROSPA
Random Number, Generation of	RANDOM
Rank Order Correlations	RANK
Reciprocal Path Analysis	PATH
Recoding Variables	TRANSFORMATIONS
Reduced Form Estimates	ECONOMETRIC
Reduced Form Predicted Values	ECONOMETRIC

Statistical Technique or Estimate	SOUPAC Source
Reduced Form Residuals	ECONOMETRIC
Reference Vector Structure	OBLIMAX, BINORMAMIN
Regression Analysis	CORRELATION, K-CLASS, MULTIPLE, STEP-WISE, EXACT, STOCHASTIC
Regression Covariance	MULTIPLE
Residual Analysis	ECONOMETRIC
Rotation of Factors	BINORMAMIN, OBLIMAX, VARIMAX, PROCRUSTES
Sample Size	CORRELATION, DISCRIMINANT, FREQUENCY, MISSING, MULTIPLE, STANDARD
Serial Correlations	AUTOCORRELATIONS
Spearman's Rho	RANK
Spearman's C	FREQUENCY
Spectrum Analysis	AUTOCORRELATIONS, CROSPA
Square Root Factors	SQUARE
Standard Deviations	CORRELATION, MULTIPLE, STANDARD, T-TEST, K-CLASS, EXACT, STOCHASTIC
Standard Error of Estimates	K-CLASS, EXACT, STOCHASTIC
Standard Errors of Regression Coefficients	K-CLASS, MULTIPLE, STEP-WISE, EXACT, STOCHASTIC
Standardized Data	STANDARD SCORES
Standardized Regression Coefficients	CANONICAL, MULTIPLE, STEP-WISE
Stepdown F	UMAVAC*
Stepwise Maximum Likelihood Factor Analysis	ITERATIVE, UNRESTRICTED
Stepwise Multiple Regression	STEP-WISE
Stochastic Restricted Least Squares	STOCHASTIC
Subject Space	POINTS OF VIEW
Sum of Squares	K-CLASS, EXACT, STOCHASTIC
Systems of Equations	
T-Statistic (Student's)	CORRELATION, MULTIPLE, STEP-WISE, T-TEST K-CLASS, EXACT, STOCHASTIC
Tchuprow's T	FREQUENCY

Statistical Technique or Estimate	SOUPAC Source
Tetrachoric Correlations	PARTIAL
Theil Compatibility Statistic	STOCHASTIC
Three Mode Factor Analysis	THREE MODE
Three Stage Least Squares	THREE STAGE
Transformation Matrix	BINORMAMIN, OLBIMAX, VARIMAX, VARISIM, PROCRUSTES
Transportation Allocations	TRANSPORTATION
Tucker-Messick Decomposition	POINTS OF VIEW
Tukey-Hanning weights	AUTOCORRELATIONS, CROSPA
Two Stage Least Squares	K-CLASS
Unrestricted Maximum Likelihood Factor Analysis	UNRESTRICTED
Variable Transformations	TRANSFORMATIONS
Variance (See Standard Deviations)	
Variance, Analysis of (See Analysis of Variance)	

INTRODUCTION TO SOUPAC

A User's Guide

I. The SOUPAC System

The University of Illinois SOUPAC system consists of (1) a library of statistical data processing programs residing in the University of Illinois Department of Computer Science IBM 360 system, (2) programmed procedures necessary to communicate among various data storage devices, the data processing programs, and SOUPAC library subroutines, (3) a special program called the syntax interpreter which translates the instructions on SOUPAC program cards into instructions to the IBM 360 computer.

To use the SOUPAC system, appropriate commands must be issued to the computer by means of a program card deck. If the user's data is on punched cards, in contrast to disk or tape, then these data cards are submitted with the program deck.

The program deck is made up of (1) Job Control Language cards, usually referred to as JCL cards and sometimes called 360 system cards, which tell the computer under which problem specification number (PS number) the program is to be run and that the SOUPAC system is to be activated, and which may also communicate additional information, especially with respect to special peripheral storage device requirements, to the computer; (2) SOUPAC statement cards which, with a few exceptions, contain the names of the SOUPAC programs being used for data processing in the job and the suboperations, if any, particular to the programs. Some of the statement cards require program constants, or parameters, to be assigned by the user to suit his own particular analysis.

Example 1 shows a typical SOUPAC job. Each line of Example 1 represents a separate punched card. Each letter or character on a given line represents a separate column on the punched card, with the leftmost letter or character

representing column one. The following representation of a card deck will be used thoughout this manual.

Example 1.

A Complete SOUPAC Job

```
/*ID  PS=9214,DEPT=PSYCH,NAME=SMITH
//  EXEC  SOUPAC
//SYSIN  DD  *
CORREL (CARDS)(PRINT)(S1/PRINT)
PRINC (S1)(S2/P)( )(100)(1).
VARIM (S2)(PRINT).
ENDS
DATA (4)(4F2.0)
 1 4 2 0
-6 3 0 1
 2 7 8 3
 0 0-1 4
 2 3 2 8
 1 5 6 2
 8 9 1 1
 1 2 2 7
 7 0 6 0
 4 7-6-2
-1 4-2 0
 8 3 4 6
END#
/*
```

The JCL statements are easily identified by either /* or // in the first two card columns. In fact, any card which has /* or // in the first two columns is considered to be a JCL card whether it was mean to be one or not. In Example 1, the first three cards and the last card of the deck are JCL cards. All remaining cards are either SOUPAC statements or data. The next three cards initiate the SOUPAC CORRELATIONS, PRINCIPAL AXIS, and VARIMAX programs in that order. The quantities in parentheses are the parameters for the respective programs. (The collection of parameters for an individual program is called the parameter string for that program.) The ENDS card, exactly one of which is always present in a SOUPAC job, signals to the syntax interpreter program that the next cards, if any, comprise a data deck. The DATA card indicates to the computer how the data is punched on the next cards. The END# card is a signal that the end of the data deck has been reached. The

last card is the end of file card which signals the computer that the end of this job has been reached.

In the program in Example 1, as the reader will recognize after reading subsequent sections, the output from the CORRELATIONS program is used internally and directly as input to PRINCIPAL AXIS, and similarly the output from PRINCIPAL AXIS is the input to VARIMAX. The capability of running a series of programs in sequence in this way in one SOUPAC job is one of the great powers of the SOUPAC system.

II. More Information About Program Cards

A. The JCL or 360 system cards

The JCL cards are needed to run all IBM 360 jobs, not just SOUPAC jobs, and are thus necessarily sometimes very complex. However, a SOUPAC job can usually be run with the four cards discussed in greater detail below and illustrated in Example 1.

1. ID card

This card has the form

```
/*ID   [ accounting information ]
```

where the acounting information includes the PS number identifying the account against which the cost of the computer run is to be charged, the department code associated with the account, the user's name, and possibly some other information. This is always the first program card.

Since ID card information requirements may change with the frequent changes in the overall IBM 360 system configuration, the user is urged to keep up to date on these requirements, and he should address any questions about ID card makeup to the SOUPAC consultants.

A user who does not have an account with the Computer Services Office will need to apply for one.

2. EXEC card

This card has the form

```
//  EXEC  SOUP
      or
//  EXEC  SOUPAC
```

where the user designates SOUP to invoke 5 intermediate storage devices and SOUPAC to invoke 15 (See page 2B.) In some cases it will be necessary to punch additional information on the EXEC card (See page 1A .)

3. SYSIN card

This card has the form

//SYSIN DD *

and communicates to the computer that the cards following are SOUPAC statement cards and data cards.

4. End of file card

This card has just /* punched in columns 1-2. It signals the end of the complete job.

Additional JCL cards are needed in special cases, as when the input comes from a user-supplied disk or tape rather than cards. A user who is not an experienced programmer will require the assistance of a SOUPAC consultant in such cases.

B. SOUPAC statement cards

There are three types of SOUPAC statements: the program parameter and subparameter cards which are the basic SOUPAC statement cards, $ control cards, and # control cards and prolog cards.

The latter two types of rather specialized cards are discussed in the section Options to a SOUPAC Job. The individual program writeups in this manual describe how the particular program parameter cards are to be punched, but a few general rules will be indicated here.

1. In general, program parameter and subparameter cards begin with a three character code (mnemonic) which is usually the first three letters of the program name or subparameter option name. Note that in Example 1 the parameter cards have CORREL, PRINC, and VARIM. In each case, all that is required is COR, PRI, or VAR; however, it is often helpful in remembering what program is being used, if one writes out more of the program name than just the first three letters.

2. Mnemonics are followed by parameters to the program. Parameters for the programs vary from program to program; parameters to all programs are defined in this manual. See Table 1 for a list of parameter types and their delimiters.

3. All parameters are "set off" by delimiters with different parameter types being set off by different delimiters. In Example 1, each program parameter is enclosed in a parenthesis pair, e.g. ( ). The CORREL statement in Example 1 has three parameters, the PRINC statement has 5 parameters, and the VARIM statement has 2 parameters.

4. A period, sometimes called the terminal delimiter, must always be punched after the last parameter used.

5. Blanks may be used freely to improve readability, and you may punch out to column 80.

6. Since all parameters are clearly separated by delimiters, any comments desired may be put before, between or after parameters. For example, the following is a valid SOUPAC control statement:

   CORRELATIONS OF RELIGIOUS ATTITUDE SURVEY WITH INPUT FROM (CARDS) AND OUTPUT TO (PRINT) AND (PRINT).

   and is equivalent to:

   COR(CARDS)(PRINT)(PRINT).

Parameter Types and Their Delimiters

Type	Delimiters	Examples
Address	( )	(S1) (P)
Integer (also called fixed point constant)	( )	(1) (20)
Index set - integer	( )	(10,20)
Real (also called floating point constant)	* *	*4.0**-0.*
Index set - real	* *	*1.,20.,2.*
Labels and character strings	" "	"ALPHA"

7. Some SOUPAC programs have subparameter options. Common examples of programs which do and do not have subparameter statements are:

Subparameter statements	No subparameter statements
BALANOVA	AUTOCORRELATIONS
FREQUENCY	BISERIAL CORRELATIONS
MATRIX	CORRELATIONS
TRANSFORMATIONS	OBLIMAX
and others	PRINCIPAL AXIS
	RANK ORDER
	STANDARD SCORES
	STEPWISE MULTIPLE CORRELATION
	T-TEST
	VARIMAX
	and others

Programs which have subparameter statements must be terminated by an ENDP card; programs which do not have subparameter statements must not be terminated by an ENDP card.

Example 1 shows three programs which must not have ENDP cards. Example 2 shows two programs which must have ENDP cards. Within the TRANSFORMATIONS program, in Example 2, we have used three subparameter statements - ADD, DIVIDE, and OUTPUT. Within the MATRIX program in Example 2, we have used two subparameter statements - MOVE and MULTIPLY. Specific descriptions of the subparameter statements can be found in the respective individual program descriptions

Example 2.

Sample SOUPAC Statements, Subparameter Statements and ENDP Card

```
TRA(CARDS).
ADD(8)(9)(8).
DIV(8)*2*(8).
OUT(S1)(1,8).
ENDP
MAT.
MOVE(CARDS(S2)).
MUL(S1)(S2)(S3/P(F)).
ENDP
```

C. The Interaction of SOUPAC Programs

The point of running a SOUPAC job is to input a set of numbers and receive as output a set of answers. Since each research problem is unique, it is usually the case that no single SOUPAC program will do the complete analysis required. Example 1 is a classic example of how output from one SOUPAC program is used as input to another SOUPAC program. Data input and output to programs is in the form of data matrices. A data matrix is a rectangular array of data; the terms row and column of a data matrix have the obvious intuitive meanings. Data matrices, when used as input or output to a program, are references by use of an address, one of the parameter types listed in Table 1.

By reading the description in this manual of the CORRELATIONS program, we find that the third parameter is the output address of the Pearson product-moment correlation matrix. By reading the description of the PRINCIPAL AXIS program we find that the first parameter is the input address and the second

parameter is an output address. Similarly, in the VARIMAX program, the first parameter is an input address and the second parameter is an output address.

We see in Example 1 that the output address of CORRELATIONS is used as the input address to PRINCIPAL AXIS and that the output of PRINCIPAL AXIS is used as input to VARIMAX. In this manner we process the raw data, step by step, to derive PRINCIPAL AXIS factors rotated according to the VARIMAX criterion.

Notice that implicit in a set of SOUPAC statements is an order in which calculations are performed. In Example 1, the CORRELATIONS program is executed first, then PRINCIPAL AXIS, then VARIMAX. Understanding this principle is essential for understanding how SOUPAC performs input and output of data matrices. In Example 1 we saw data matrices passed from program to program by being stored on intermediate storage, labeled S1, S2, and S3.

Example 3a.

Input of One Card Data Deck

```
MATRIX
MOVE(CARDS)(S1).
ENDP
CORREL(S1)(PRINT)(S3/PRINT).
STANDARD SCORES(S1)(S2/PRINT)(PRINT)(1).
```

Example 3b.

Input of Two Card Data Decks

```
CORREL(CARDS)(PRINT)(S3/PRINT).
STANDARD SCORES(CARDS)(S2/PRINT)(PRINT)(1).
```

Example 3 points out a critical detail in the proper use of addresses. In Example 3a one card deck is read and stored as a data matrix on S1 by use of the MATRIX MOVE suboperation. This matrix is then used as input to both CORRELATIONS and STANDARD SCORES. Notice that using S1 as input to CORRELATIONS does not destroy the contents of S1 and that the data input to STANDARD

SCORES is the same as though CORRELATIONS had not been executed. As long as an address is used only for input, the matrix stored at that address remains unchanged.

Example 3b is similar to Example 3a in that the input address to CORRELATIONS and STANDARD SCORES is again the same; however, in Example 3b this address is now CARDS, not S1.

Using CARDS as an address differs from using an intermediate storage file such as S1 or S2 in that subsequent uses of CARDS mean "read the next card data deck."

Therefore, Example 3b will read two data card decks, the first one being read by CORRELATIONS and the second one being read by STANDARD SCORES. Consecutive uses of an S-type address for input will cause the same data to be read from that file each time; consecutive uses of CARDS for input will cause the next card data deck to be read each time.

For a more complete discussion of SOUPAC addresses, read the section, SOUPAC Input/Output and Intermediate Storage.

D. Placement of Data Decks

Whenever CARDS is used as an input address, a data deck will be read. Each data deck must be preceded by a data format card and followed by an END# card. If more than one data deck is required, as would be the case in Example 3b, the decks are placed in the order they will be read by the SOUPAC control statements. See Example 4 for a SOUPAC job that uses more than one data deck. Recall that there is one ENDS card in a SOUPAC job and its primary function is to separate the SOUPAC statements from the data card decks.

Example 4.

SOUPAC Job With Two Card Data Decks

```
/*ID PS=9214,DEPT=PSYCH,NAME=SMITH
//  EXEC  SOUPAC
//SYSIN  DD  *
CORREL (CARDS) (PRINT) (S3/PRINT).
STANDARD SCORES (CARDS) (S2/PRINT) (PRINT) (1).
ENDS
DATA(10)(10F6.2)
  .
  .
  .   [data deck for CORRELATIONS Program]
  .
  .
END#
DATA(8)(8F10.3)
  .
  .
  .   [data deck for STANDARD SCORES Program]
  .
  .
END#
/*
```

E. The Data Matrix

It is common practice to represent statistical data as a rectangular array of numbers called a data matrix. This convention is quite natural since data samples frequently are observations taken over sets of variables. For example, suppose we have collected data on twenty people by asking them their sex, age, height, weight, and income. If we wrote this information in tabular form on a piece of paper, we would probably have six columns; one column for the subjects' names, and one column each for each of the five variables: sex, age, weight, height and income. There would be twenty entries in this table, one for each subject. What we have then in tabular form is a rectangular representation of our data sample with the variables, including the name, as columns, and the observations as rows.

From this representation it is a short step to punching the table onto cards, one card per person. If there are more variables per observation than will fit on one card, we use as many consecutive cards as are necessary. To make the job of setting up a data matrix for reading as simple as possible, the SOUPAC user should always follows these rules:

1. Begin each observation on a new card.
2. If there is more than one card per observation do not "split" a multi-column variable across a card boundary. For example, if income takes six columns, and there are three columns left on the current card, go to a new card. Do not put the first three columns of income on the end of the current card, and the last three columns on a new card.
3. Always have the number of cards per observation constant within a given data deck.
4. Always punch variables in identical card columns for each observation. Each observation of a given data deck should have the same card and card column organization.

F. The Data Format Card and the END# Card

Whenever CARDS is used as an input address, the SOUPAC input routines must know how the input data matrix is represented on punched cards. In particular, they must know:

1. What are the dimensions (number of rows and columns) of the data matrix being represented.
2. Which card columns represent which variables.
3. How many cards per observation (row) are there.

The function of the data format card and the END# card is to provide precisely this information.

Data format cards have the following form:

(FA([number of variables]) ([format])

For example, Example 1 had the data format card:

(FA(4) (4F2.0)

Notice that the number of columns (variables) of the data matrix must be explicitly indicated to the computer. The format is used to

describe how one observation is punched. All observations are assumed to have the same card column structure so that the computer is able to determine the number of rows of the data matrix by simply reading cards under control of the format provided until it finds the END# card. Since the format defines one observation, implicit in this definition is how many cards per observation are used.

The format is a FORTRAN type format and follows all the rules of FORTRAN with the added restriction that it may not be longer than 592 characters. If the format itself does not fit on one card, one simply continues punching the format onto the next card. No continuation marks are required since the computer stops reading format cards when it finds the right parenthesis which closes the format string.

It is possible to write very complex formats, and the user who wishes to explore such possibilities is referred to any standard FORTRAN language text. For most uses, however, the X and F field specifications, discussed in detail below, will be quite adequate.

It is conventional in computer programming to refer to a particular group of contiguous columns on a data card as a field. Hence a four digit number will occupy a field of at least four columns when punched on a data card, and the field will be larger if an algebraic sign and/or a decimal point is also punched. The field specifications in a format tell the computer which fields are assigned to the variables being input and describes how the numbers are to be interpreted. In particular:

1. The field specification nX, where n is an integer, tells the input routine to skip n columns. These n columns are said to be a skip field.

2. The field specification Fw.d, where w and d are integers, and $w > d$, specifies that a field of w columns is to contain a decimal number and that in case the decimal point is not actually punched then the number is to be

divided by 10 to the d power before being stored in the computer. However, if the decimal point is actually punched then it overrides the d indicator. (Remember - w must include a column for the decimal point!) For example, the field specification F7.2 means that the corresponding field of 7 columns contains a decimal number which is to be divided by 10^2, or 100, unless a decimal point is punched.

The user should be warned that any blanks in a field read by F field specification are read as zeroes! Suppose that card columns 6-10 are read according to F5.1 and 572 is punched in columns 6-8, then the number stored by the computer is 57200/10 or 5720.

The field specifications must be separated by commas, but a sequence such as F6.3, F6.3, T6.3, F6.3 can be represented by 4F6.3, which saves space and keypunching labor. This representation is used in Example 1 where DATA(4)(4F2.0) is punched instead of DATA(4)(F2.0,F2.0,F2.0,F2.0).

Data fields on the data cards are represented left to right in the format. Example 5a indicates that we have ten variables and that these ten variables are punched in consecutive five column fields beginning in column sixteen. Notice that we have defined only the last 05 of 80 card columns. The first 15 columns will simply be ignored.

If an observation goes over one card, use the / (slash) to indicate that the data fields continue on the next card.

Hint: The number of data cards considered to be on observation is the number of slashes in the format plus one, i.e. no slashes implies one card per observation, one slash implies two cards per observation.

With minimum practice, format reading becomes almost automatic. Example 5b can be read as follows:

1. There are twenty variables in this data matrix.
2. a. Beginning in card column one, skip five columns.
   b. Read one six column variable.
   c. Skip two card columns.
   d. Read one three column variable.
   e. Skip four card columns.
   f. Read nine variables of six columns each.
   g. Go to a second card (ignoring the last six columns of the current card).
   h. Skip the first twenty columns of the second card.
   i. Read nine variables of six columns each.

Always remember that there are 80 columns on a punched card. If a format specifies a card which has more than 80 card columns, an error will result. Example 5c shows a format which specifies 90 columns for the first card of a two-card observation. The use of this card will result in an error.

The number of variables should agree with the number of field specifications, including replications, in the format. Experienced programmers can sometimes take exception to this rule in order to shorten format length, but it is a dangerous practice, and the discrepancy between these two counts is one of the more frequent errors made in punching the data format card.

Example 5a.
Sample Data Format Card

```
DATA (10) (15X,10F5.0)
```

Example 5b.
Data Format Card Indicating Two Cards Per Observation

```
DATA(20)(5X,F6.0,2X,F3.0,4X,9F6.0/20X,9F6.0)
```

Example 5c.
Invalid Data Format Card - More Than 80 Columns Specified

```
DATA(20)(10X,10F8.0)
```

Example 5d.
More Variables Specified Than Data Fields

```
DATA(21)(5X,F6.0,2X,F3.0,4X,9F6.0/20X,9F6.0)
```

Once a data deck has been read into SOUPAC, all information concerning which card columns contained which variables is lost. Within SOUPAC a data sample is handled as a data matrix and variables are referenced by variable number, not by card column.

G. Keypunching Hints

1. Whenever possible code variables using numbers instead of letters. For example, code sex as 0, 1 rather than F, M.

2. Avoid confusing O, the letter oh, and 0, the number zero. Oh and zero have different hole codes in the punched card. To help avoid possible confusion, it is common to see the letter written as Ø. Unless the meaning of O is clear from the context in which it is used, assume that Ø means the letter and 0 means the number (Warning, other computer installations often disagree on this convention and represent zero by Ø).

3. Code missing data as blanks (no punches). Some researchers code missing data as 9, 99, or 999 depending upon the number of card columns which the variable uses. This method has the disadvantage that once the data cards have been read the variables are treated as matrix columns, and missing data values are not coded consistently for all variables. Also, those SOUPAC programs which correct for missing data presume that missing data has been coded as a blank. Within the computer a blank is represented as -0 (minus zero).

4. Punched cards which are warped or ragged-edged are likely to be rejected, or perhaps misread, by the card reader, and such cards should be discarded before they cause problems. The /*ID card in front of the deck and the /* card at the end of the deck are especially susceptible to rapid wear.

5. Overpunching is the attempt, usually unsuccessful, to punch more information in one column of a card than is normally allowed. The result is often undefined hole combinations which the card reader rejects as a "read error." Avoid overpunching.

Programming errors are difficult to avoid in writing even simple programs. Skill in the detection and correcting of errors, or "debugging," is an important facet of the programming art. The occasional SOUPAC user usually

does not acquire the experience necessary to become adept at detecting errors, but there are a number of things that he or she can do to minimize them and detect the more obvious ones.

One of the most simple and important error saving strategies is one that is seldom used - avoid rushing the job! Several days should be allowed for writing and debugging short programs and as much as several weeks for complex ones. Even when debugging the program is a minor problem, unexpectedly long turn-around times due to heavy job volumes or computer breakdowns can foul tight schedules. Long range planning is a key part of data processing.

Programming errors can be broadly categorized into two types, depending on whether they disrupt the translation of the program into machine instructions ("compile-time" errors) or whether they disrupt the computer's actual execution of the machine instructions ("execution-time" errors). Occurrence of either type error will normally cause the computer to print an appropriate error message for diagnostic purposes. The user will usually find that compile-time errors, which are most often caused by mispunching or deleting characters, are relatively easy to find. However, execution-time errors are often due to incorrectly punching data or using an incorrect format statement, resulting in unexpected numbers being processed, and considerable time and ingenuity may be needed to run such an error to ground.

Here are some debugging hints:

1. Obviously the program deck should be carefully checked for keypunching errors before it is submitted to be run.

2. Make sure that the cards are in the correct order.

3. Missing ENDS and ENDP cards are a common source of error. Check for these.

4. After correcting an error for which an error message has been explicitly generated by the computer, carefully reexamine the entire program.

The computer may not detect all errors on the same run, and you may find one which the computer has not yet seen but which would abort the next run.

5. Make sure that the parameters specified on the /*ID card are such that the job can run to completion.

6. TAKE YOUR TIME!

DATA AND MATRIX MANIPULATIONS PACKAGE

The data manipulative programs

As implied by the name, TRANSFORMATIONS is used for performing data transformations often necessary in "setting up" data to be input to one of the "cook-book" programs of the SOUPAC system. The user may create new variables as linear combinations or as algebraic functions of old variables, and the user may recode or alter data values on the basis of test conditions by using the TRANSFORMATIONS program. To perform matrix algebra operations with one or more matrices, to augment matrices either row-wise or column-wise, to reorder, save or delete specific rows or columns of a matrix, one would use the MATRIX program. MATRIX also has the capability of printing, punching card decks, and reading and writing tape and disk files in ways not available elsewhere in SOUPAC.

The TRANSFORMATIONS and MATRIX programs have a unique place in the SOUPAC system. Although the remainder of the SOUPAC library performs a large number of specialized statistical procedures, there are some computations which are not represented by a uniquely written program. However, by an imaginative utilization of the combined powers of TRANSFORMATIONS and MATRIX it is possible to perform a virtually unlimited range of established and experimental statistical techniques. It is a common practice within the SOUPAC office to "check out" newly written programs against results computed by TRANSFORMATIONS and MATRIX. Conversely, these two programs can be used as teaching tools by having students learn the step-by-step computations and then checking the results against results of the "cook-book" programs. Complete multiple regressions and analysis of variance programs, for example, have been written in this manner. Since most SOUPAC jobs require the use of TRANSFORMATIONS and MATRIX, a familiarity with these two programs is basic to an effective use of the SOUPAC system.

MATRIX

I. General Description

The MATRIX program is a data manipulating program for inputting and outputting, creating, performing matrix algebraic operations on, and generally handling data matrices. All the MATRIX suboperations are restricted to 1000 columns (variables). No absolute limit is set on the number of rows (observations).

Standard SOUPAC address conventions are used including the use of the character X to denote punched output, and (F) after a print to denote print with F format. Also available and discussed in section III below is the use of I for storing a matrix in memory, and the use of (L) after a print to invoke the MATRIX labeling feature. All other restrictions are noted by the discussion of the individual suboperation explanations.

II. Parameters

A. Main Parameters

The MATRIX program is invoked by coding the name MATRIX, or simply the mnemonic MAT on a program card. There are two optional parameters available which may be coded on the MAT card.

1. If it is desired that the program print, immediately prior to the execution of each MATRIX subparameter operation, the time in seconds since entry into the MATRIX program, code a (1) after the name MATRIX. This option is not normally needed and is provided merely for giving timing estimates. Example:

   MATRIX (1).

2. The second optional parameter is coded as a (1) following the timing estimate parameter. This second option is used to <u>suppress</u> printing by the program of the number of rows and columns and the precision of all answers, i.e. output, matrices. Normally, the program will always print out this information. Example:

   MATRIX ()(1).

   If both options one and two are desired use:

   MATRIX (1)(1).

   If neither option is desired, as is generally the case, use:

   MATRIX.

Note that as in all SOUPAC programs, the main program statement, and also each subprogram statement, must be terminated by a period.

B. Subparameters

Any MATRIX operation may be invoked by coding its mnemonic followed by appropriate subparameters. All operations in MATRIX handle both single and double precision matrices at the control of the user (see operations SINGLE and DOUBLE). For an address not explicitly assigned either single or double precision, MATRIX assumes a default of double precision for output to the address. Terminate all subparameter statements with a period.

To end a MATRIX program, place a card which has the characters END P after the last MATRIX subparameter card. Since all MATRIX programs must have at least one subparameter operation, an error will be signaled if a MAT card is followed immediately by an END P card.

Input and output for MATRIX may be from any source, however, the following rules must be observed:

1) Never use CARDS as input to any operation except MOVE unless both the number of rows and the number of columns have been specified on the DATA format card at the front of the data deck.

2) You may never have an output address of only (PRINT) or (P). All MATRIX output must go to some intermediate storage location even when only printout is desired; for example (S1/P).

3) Avoid using the same address more than once on the same parameter card unless otherwise noted in the description of an individual operation. However, in those operations which do permit using an address more than once as an input address, CARDS may not be used as an input address more than once. In all operations except INVERT, never specify an output address which is the same as an input address for that operation.

4) The contents of an input address remains unchanged during the execution of an operation unless otherwise noted.

Following is a description of the subparameter operations currently in the MATRIX program.

Summary of MATRIX Operations

mnemonic	notes	operation name	examples
ABS		Absolute value	ABS (S1) (S2).
ADD	1,2,5	Add	ADD (S1) (S2) (S3).
ALL	3,6	All	ALL (S1) "GT" *0* (S2). ALL (S1) "NE" *-0* (S2) (2). ALL (S1) "LE" (S2) (S3) (1,10,3).
ANY	3,6	Any	ANY (S1) "GE" (S2) (S3). ANY (S1) "EQ" *1.* (S2) (1,2) (5). ANY (S1) "LT" *99* (S2).
CHO	4	Cholesky decomposition	CHO (S1) (S2).
COL	6	Column delete	COL (S1) (S2) (2). COL (S1) (S2) (10) (12,15).
CON		Constant addition	CON (S1) *2.* (S2). CON (S1) (S2) (S3).
COU		Count	COU (S1) (S2). COU (S1) (S2) (1). COU (S1) (S2) (2).
CRO		Cross product	CRO (S1) (S2). CRO (S1) (S2) (1).
DIA		Diagonal to vector	DIA (S1) (S2).
DIM		Dimension	DIM (S1) (S2).
DOU	2	Double precision	DOU (S1).
EJE		Eject	EJE.
E-D	1,2,5	Elementwise divide	E-D (S1) (S2) (S3).
E-M	1,2,5	Elementwise multiply	E-M (S1) (S2) (S3).
E-R		Elementwise square root	E-R (S1) (S2).
EXP		Expand	EXP (S1) *20* (S2). EXP (S1) (S2) (S3).
FIL	2	File	FIL (S1).
GEN	6	Generate	GEN (S1) *1*.
HOR	1,2,5	Horizontal augment	HOR (S1) (S2) (S3).
IDE		Identity matrix	IDE (48) (S1).
INP		Input	INP (S1) (S2) ( ) (9) "(9F6.1)". INP (S1) (S2) ( ) (12). INP (S1) (I) (576) (10).
INV	4	Invert	INV (S1) (S2). INV (I) (I). INV (S1) (S2) (1) (1) *10.E-6*.
KRO		Kronecker product	KRO (S1) (S2) (S3).
LAB	6	Label	LAB (S1) "SAMPLE 1" "AGE" "SEX".
LAG		Lag	LAG (S1) (2) (6) ( ) (S2). LAG (S1) (3) (4) (1) (S2).
LOW		Lower triangle	LOW (S1) (S2). LOW (S1) (S2) (1).

mnemonic	notes	operation name	examples
MAX		Maximum value	MAX (S1) (S2). MAX (S1) (S2) (1). MAX (S1) (S2) (2).
MIN		Minimum value	MIN (S1) (S2). MIN (S1) (S2) (1). MIN (S1) (S2) (2).
MOV	2	Move	MOV (C) (S1). MOV (S1) (S2).
MUL	1,4,5	Multiply	MUL (S1) (S2) (S3).
OUT		Output	OUT (S1) (S2) "(10E16.9)". OUT (S1) (S2).
PAR		Partition	PAR (S1) (S2) (5) (10) (2) (21).
PER	6	Permutation	PER (S1) (2) (1).
PRI		Print	PRI (S1) "(' ',10F13.4)".
PUN		Punch	PUN (S1) "(8F10.3)".
REC		Reciprocal	REC (S1) (S2).
REM		Remap	REM (S1) (S2) (8).
REW	2	Rewind	REW (S1).
ROW	6	Row delete	ROW (S1) (S2) (1) (2) (3). ROW (S1) (S2) (2,20,3) (3,20,3).
RSD		Reciprocal of Square Root of Diagonal	RSD (S1)(S3).
SCA		Scalar multiply	SCA (S1) *.1* (S2). SCA (S1) (S2) (S3).
SIN	2	Single precision	SIN (S1).
SUB	1,2,5	Subtract	SUB (S1) (S2) (S3).
SUM		Sum	SUM (S1) (S2). SUM (S1) (S2) (1). SUM (S1) (S2) (2).
TRA	4	Transpose	TRA (S1) (S2).
UPP		Upper triangle	UPP (S1) (S2). UPP (S1) (S2) (1).
VEC		Vector to diagonal	VEC (S1) (S2).
VER	1,2,5	Vertical augment	VER (S1) (S2) (S3).

1. Conformability of input matrices is checked.
2. Up to twenty-one total addresses may be used.
3. A warning message is printed if no rows are output.
4. Any matrix which has been previously stored under the I address will be destroyed.
5. An input address may be used more than once for input to the same instruction.
6. As many arguments as are needed of the last argument type may be used.

ABSOLUTE VALUE (mnemonic: ABS)

The ABSOLUTE VALUE operation has two address parameters, an input address and an output address. The absolute value of each element of the input matrix is taken and the result goes to the output address. Example:

```
ABSOLUTE VALUE (SEQ3)(SEQ4).
```

ADD (mnemonic: ADD)

The ADD operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of rows and columns as all other input matrices. An address may be used more than once as an input address.

Corresponding elements of the first matrix through the next to last matrix are added together, and the result goes to the output address. Examples:

```
ADD (SEQ1)(SEQ4)(SEQ5).
ADD (SEQ1)(SEQ4)(SEQ2)(SEQ3)(SEQ5).
```

ALL (mnemonic: ALL)

The ALL operation performs a particular test, specified by the second operand as a relational operator, between a set of elements for each input row and a floating point number specified by the third operand. If all elements of the set for a given row pass the test, that row is output to the output address.

The first parameter is the input address. The relational operator is enclosed in quotation marks. The third operand may be either a floating point number or an address in which case the first element of the matrix is used as the floating point number. The six legal relational operators are "LT", "LE", "EQ", "NE", "GT", and "GE". Remaining (optional) parameters are index sets specifying which variables are to be included in the testing. If no variables are specified, all variables are included in the testing. Note that if only one variable is specified, the results of the ANY and the ALL operation would be the same. Examples:

```
ALL (SEQ1) "NE" *0.* (SEQ2).
ALL (SEQ3) "GE" (SEQ4)(SEQ1).
```

ANY (mnemonic: ANY)

The ANY operation performs a particular test, specified by the second operand as a relational operator, between a set of elements for each input row and a floating point number specified by the third operand. If any element of the set for a given row passes the test, that row is output to the output address.

The first parameter is the input address. The relational operator is enclosed in quotation marks. The third operand may be either a floating point number or an address in which case the first element of the matrix is used as the floating point number. The six legal relational operators are "LT", "LE", "EQ", "NE", "GT", and "GE". Remaining (optional) parameters are index sets specifying which variables are to be included in the testing. If no variables are specified, all variables are included in the testing. Note that if only one variable is specified, the results of the ANY and the ALL operation would be the same. Examples:

```
ANY (SEQ2) "GT" *3.* (SEQ1).
ANY (SEQ1) "NE" *-0.* (SEQ4)(1,3).
```

CHOLESKY (Mnemonic: CHO)

The CHOLESKY operation decomposes a square symmetric matrix into the product of an upper triangular matrix and a lower triangular matrix such that the two triangular matrices are the transpose of each other. This method of decomposition is sometimes called the "square root" method. CHOLESKY has two operands, an input address and an output address. The result which goes to the output address is the lower triangular matrix resulting from the decomposition.

If the input matrix is not square, the "extra" rows or columns are ignored. Additionally, if the square matrix is not symmetric, the actual upper triangle of the matrix is effectively ignored and is instead assumed to be identical to the lower triangle. Example:

```
CHOLESKY (SEQ1)(SEQ2).
```

If the input matrix on SEQ 1 is

```
 4    -2    -4
-2     2     3
-4     3     6
```

the resulting matrix output to SEQ 2 is

```
 2     0     0
-1     1     0
-2     1     1
```

COLUMN DELETE (mnemonic: COL)

The COLUMN DELETE operation specifies which columns of an input matrix are to be deleted before sending the result to the output address. The first parameter is the input address. The second parameter is the output address. Columns to be deleted are specified by index sets following the output address. Examples:

```
COLUMN DELETE (SEQ4)(SEQ5)(4)(5)(6)(8)(13)(15).
COLUMN DELETE (SEQ1)(SEQ2)(1,30,3).
```

CONSTANT (mnemonic: CON)

The CONSTANT operation has three parameters. A floating point number specified by the second operand is added to every element of the matrix specified by the first operand. The result goes to the third operand address.

The second operand may be either a floating point number enclosed in asterisks or a standard SOUPAC input address. If an address is specified, the first element of the matrix at the address is used for the floating point number. Examples:

```
CONSTANT (SEQ1) *4.5* (SEQ2).
CONSTANT (SEQ1)(SEQ3)(SEQ4).
```

COUNT (mnemonic: COU)

The COUNT operation has three operands; an input address, an output address, and an option indicator.

If option 0 is specified, the resulting output matrix is a single row vector containing a count of the number of elements, excluding missing data, of each column of the input matrix. Specifying no option is equivalent to specifying option 0.

If option 1 is specified, the resulting output matrix is a single column vector containing a count of the number of elements, excluding missing data, of each row of the input matrix.

If the option is specified as any number other than 0 or 1, a single element matrix is output which contains a count of the number of elements, excluding missing data over the entire matrix. Examples:

```
COUNT (SEQ1)(SEQ4).
COUNT (SEQ5)(SEQ3)(1).
COUNT (SEQ2)(SEQ3)(2).
```

CROSS PRODUCT (mnemonic: CRO)

The CROSS PRODUCT operation has three operands, an input address, an output address, and one option flag. The output address is the square symmetric matrix

$$X^TX$$

which results from the matrix multiplication of X^T and X, where X is the input matrix and X^T is the transpose of the input matrix.

The option flag, coded as (1), is used whenever it is desired that the X matrix used in forming the cross products matrix is the input matrix with an additional column of 1's as the first column of the matrix. Note that when the option is used, the output matrix will have one more row and column than if the option is not used.

DIAGONAL (mnemonic: DIA)

The DIAGONAL operation has two operands, an input address and an output address. The main diagonal elements of the first matrix are used to form a single row vector which is output to the second operand address. Example:

DIAGONAL (SEQ2)(SEQ4).

DIMENSION (mnemonic: DIM)

The DIMENSION operation has two address parameters, an input address, and an output address. The number of rows and the number of columns of the input matrix are used to form the first and second elements respectively of a two element, single row matrix which is output to the output address. Example:

DIM (SEQ2)(SEQ1).

DOUBLE (mnemonic: DOU)

The DOUBLE operation has anywhere from one to twenty-one addresses as parameters. Listing an address as a parameter negates the effect of any previous listing of that address as a parameter in the operation SINGLE. Listing an address as a parameter which has not appeared as a SINGLE subparameter has no effect. Example:

DOUBLE (SEQ1)(SEQ2)(SEQ3)(SEQ4)(SEQ5)(I).

EJECT (mnemonic: EJE)

The EJECT operation causes the next printout to begin at the top of a new page. EJECT has no parameters. Example:

EJECT.

E-DIVIDE -- Elementwise Divide -- (mnemonic: E-D)

The E-DIVIDE operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of rows and columns as all other input matrices for the use of the operation. An address may be used more than once as an input address.

Elements of the second matrix through the next to last matrix are divided into the corresponding elements of the first matrix. Output goes to the last address. Example:

E-DIVIDE (SEQ1)(SEQ2)(SEQ3).

E-MULTIPLY -- Elementwise Multiply -- (mnemonic: E-M)

The E-MULTIPLY operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same

number of rows and columns as all other input matrices for the use of the operation. An address may be used more than once as an input address.

Corresponding elements of the first matrix through the next to last matrix are multiplied together. Output goes to the last address. Examples:

```
E-MULTIPLY (SEQ1) (SEQ1) (SEQ2).
E-MULTIPLY (SEQ3) (SEQ2) (I) (SEQ4).
```

E-ROOT -- Elementwise Square Root -- (mnemonic: E-R)

The E-Root operation has two address parameters, an input address and an output address. The (positive) square root of each element of the input matrix is taken and the result goes to the output address. Example:

```
E-ROOT (SEQ1)(SEQ2).
```

EXPAND (mnemonic: EXP)

The EXPAND operation takes the first row of the first input matrix and repeatedly outputs that same row to the output address the number of times specified by the second parameter.

The second parameter can be either a floating point number in which case the input row is copied to the output address the number of times specified by the integer portion of the floating number; or the second parameter can be an input address in which case the input row is copied to the output address until the output matrix has the same number of rows as the second input matrix. The third parameter is the output address. Examples:

```
EXPAND (SEQ1)(SEQ2)(SEQ3).
EXPAND (SEQ1)*55*(SEQ4).
```

FILE (mnemonic: FIL)

The FILE operation has anywhere from one to twenty-one addresses as parameters. FILE is used to cause an end-of-file mark to be written at the end of a SEQUENTIAL file. This operation is generally most useful to the user who wishes to place more than one file on his own physical tape. Since any meaningful use of the FILE operation requires the addition of appropriate IBM 360 JCL cards, all but the most experienced users should see a consultant in the SOUPAC office before using this operation. Example:

```
FILE (SEQ5).
```

GENERATE (mnemonic: GEN)

The GENERATE operation generates a single row vector with the floating point numbers the user specifies. The first operand is the output address. Remaining parameters are as many floating point numbers as the user wishes. Example:

```
GENERATE (SEQ2) *1.* *2.* *4.* *8.* *16.* *32.*.
```

HORIZONTAL AUGMENT (mnemonic: HOR)

The HORIZONTAL AUGMENT operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of rows as all other input matrices for the use of the operation. An address may be used more than once as an input address.

Input matrices from the first matrix through the next to last matrix are stacked left to right and the result goes to the last address. Example:

HORIZONTAL AUGMENT (SEQ1)(SEQ4)(SEQ2)(I).

IDENTITY (mnemonic: IDE)

The IDENTITY operation has two parameters. An identity matrix, of order specified by a fixed point number as the first operand, is output to the address specified by the second operand. Example:

IDENTITY (20)(3F).

INPUT (mnemonic: INP)

The INPUT operation will input formatted or non-formatted records from any available device. This option is primarily for reading card images or other similar data the user may have usually on his own tape, which would be awkward to input in the typical card deck manner.

Never input to T (see SPECIAL COMMENTS) using the INPUT operation unless both the number of rows and the number of columns of the input matrix are specified as parameters on the INPUT operation parameter card.

The parameters for INPUT are the input address, the output address, the number of rows of the input matrix (optional in most cases), number of columns, and optionally the format enclosed in quotation marks. Examples:

INPUT (SEQ1)(SEQ2/PRINT)(20)(5)"(10F8.3)".
INPUT (SEQ2)(SEQ3)( )(8).

INVERT (mnemonic: INV)

The INVERT operation inverts a non-singular real matrix. The INVERT operation has five subparameters, the last three of which are optional. The first parameter is the address of the matrix to be inverted, and the second parameter is the output address of the result. (The incore address option described in section III.A - SPECIAL COMMENTS - may be used for either input, output, or both). To have the determinant of the original matrix printed out, code a (1) as the third parameter.

The inversion technique used is the Gauss-Jordan method with pivot elements assumed to be on the main diagonal. If it is desired that the inversion technique perform row and column interchange, for the purpose of picking pivot elements as those with the largest absolute value at each step of the elimination procedure,

code a (1) as the fourth parameter. The default case, pivot elements assumed to be on the main diagonal, executes faster than when row and column interchange is performed. For those real symmetric matrices which have the property that the largest elements are necessarily on the main diagonal (e.g. correlation, cross-products, variance-covariance matrices) numerical accuracy of the results is not significantly different between the two options. For general matrices in which specific properties are not known, using row and column interchange will probably produce more accurate results.

The fifth argument is a floating point number enclosed in asterisks which is to be used as the criterion for singularity. If the absolute value of any pivot element is less than the criterion for singularity, the matrix is assumed to be singular. If no value is specified, or if *0.* is specified as the fifth parameter, a default value of 10^{-8} is used to test for singularity.

INVERT destroys any previous use of the incore address option. All calculations are done in double precision.

INVERT also has the ability to solve a set of simultaneous linear equations if a unique solution exists. To solve the system indicated by the matrix equation

$$AX = Y$$

input to the INVERT suboperation a matrix which contains A:Y (i.e. the constant term appearing as the last column variables). The resulting output of the INVERT suboperation will be

$$A^{-1}:X$$

The Y above may be more than one column vector in which case each resulting column vector of X will be the solution for the corresponding column of Y. Examples:

```
INVERT (SEQ1)(SEQ2).
INVERT (SEQ4)(SEQ3)(1)(1).
INVERT (SEQ2)(SEQ4)(1)( ) *10.E-5*.
INVERT (SEQ5)(SEQ1)( )(1) *.0000001*.
```

KRONECKER PRODUCT (mnemonic: KRO)

The KRONECKER PRODUCT operation forms the Kronecker Product of two matrices and outputs the results to an output address. The resulting output matrix is composed of $m_1 \times n_1$ submatrices where m_1 and n_1 are the dimensions of the first input matrix. Each submatrix has the size $m_2 \times n_2$ where m_2 and n_2 are the dimensions of the second input matrix. Note that the output matrix has the dimensions $m_1m_2 \times n_1n_2$. Each submatrix is the result of the scalar product $a_{ij} B$.

For example, if matrix A is on S1 and B is on S2 where

$$A = \begin{matrix} 1. & 2 \\ -1 & 1 \\ 0 & .5 \end{matrix} \qquad B = \begin{matrix} 1 & 3 \\ 2 & 4 \end{matrix}$$

the result of executing the MATRIX statement

```
KRONECKER (S1)(S2)(S3).
```

would be the following matrix on S3.

1	3	2	6
2	4	4	8
-1	-3	1	2
-2	-4	3	4
0	0	.5	1.5
0	0	1	2

LABEL (mnemonic: LAB)

The LABEL operation is used to store a title and column labels at a SOUPAC address for later use within a MATRIX program. The title is limited to 128 characters. Labels are limited to eight characters each.

The first parameter is the address where the title and labels are to be stored. This is then followed by the title and labels each enclosed in quotation marks. Only one label set is active at any one time. Hence, each use of LABEL overrides all previous uses. Labels generated within a MATRIX program may not be passed to other programs. (note: The incore address option may be used to store a title and label set if desired.)

To use a set of labels which have been stored, use (L) after the print portion of the output matrix to be labelled. For example, the following statement pair will label the result of a HORIZONTAL AUGMENT.

```
LABEL(S5)"SAMPLE DATA""ID""AGE""HEIGHT""WEIGHT"
HORIZ(S1)(S2)(S3/P(L)).
```

S5 is being used as a convenient place to store the labels. This presumes that S5 is not being used for anything else and is available.

LAG (Mnemonic: LAG)

The LAG operation has the following operands; an input address, an integer specifying the number of lag periods to be added, an output address, and index sets specifying which variables are to be lagged.

For example, suppose we are interested in lagging the fourth variable of a five variable matrix and suppose that we want three lag periods. First it should be noted that the resulting output matrix will necessarily have three fewer rows (observations) than the input matrix.

If we use

```
LAG(S1)(3)(S2)(4).
```

the resulting t^{th} row of the output address would be

$$X_{t,1} \quad X_{t,2} \quad X_{t,3} \quad X_{t,4} \quad X_{t-1,4} \quad X_{t-2,4} \quad X_{t-3,4} \quad X_{t,5}$$

As a concrete example, if the following input is on S1

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15
16	17	18

Use of either the statement

```
LAG(S1)(2)(S2)(2)(3).       or
LAG(S1)(2)(S2)(2,3).
```

will result in the following matrix being output onto S2.

7	8	5	2	9	6	3
10	11	8	5	12	9	6
13	14	11	8	15	12	9
16	17	14	11	18	15	12

LOWER TRIANGLE (mnemonic: LOW)

The LOWER TRIANGLE instruction copies a matrix from one address to another and sets all elements which are above the main diagonal to zero. It is possible to indicate if it is desired that the main diagonal elements also be set to zero.

The LOWER TRIANGLE instruction has three operands; an input address, an output address, and an integer option flag. If the option flag is omitted or is zero, the main diagonal elements are included as part of the lower triangle. If the option flag is non-zero, the main diagonal elements are set to zero. Examples:

```
LOWER (S1)(S3).
LOWER (S2)(S4)(1).
```

MAXIMUM (mnemonic: MAX)

The MAXIMUM operation has three operands; an input address, an output address, and an option indicator.

If option 0 is specified, the resulting output matrix is a single row vector containing the maximum element of each column of the input matrix. Specifying no option is equivalent to specifying option 0.

If option 1 is specified, the resulting output matrix is a single column vector containing the maximum element of each row of the input matrix.

If the option is specified as any number other than 0 or 1, a single element matrix is output which contains the maximum element of the entire matrix. Examples:

```
MAX (SEQ1)(SEQ3).
MAX (SEQ1)(SEQ2)(2).
```

MINIMUM (mnemonic: MIN)

The MINIMUM operation has three operands; an input address, an output address, and an option indicator.

If option 0 is specified, the resulting output matrix is a single row vector containing the minimum element of each column of the input matrix. Specifying no option is equivalent to specifying option 0.

If option 1 is specified, the resulting output matrix is a single column vector containing the minimum element of each row of the input matrix. Examples:

```
MIN (SEQ1)(SEQ3).
MIN (SEQ1)(SEQ2)(2).
```

MOVE (mnemonic: MOV)

The MOVE operation moves (actually copies) a matrix from one SOUPAC standard input source to another. If reading from SEQUENTIAL, the MOVE operation assumes that the data set was created using SOUPAC conventions, i.e. by some SOUPAC program. If the input source is CARDS, the input deck must be preceded by a correct DATA format statement and terminated by an END# card.

Never MOVE from CARDS to I (see SPECIAL COMMENTS) unless both number of rows and number of columns of the input matrix are specified at the front of the data deck.

The operation has between two and twenty-one addresses as parameters. The first address is the input address. All remaining addresses are output addresses. Examples:

```
MOVE (CARDS)(SEQ1)(SEQ2).
MOVE (CARDS)(SEQ1).
```

MULTIPLY (mnemonic: MUL)

The MULTIPLY operation has three addresses for parameters. A matrix multiplication is performed between the matrices on the first two addresses and the result is stored in the third address. The MULTIPLY operation permits use of the same address to be used as an input address for both first and second operands. This usage is equivalent to using the SQUARE suboperation.

The incore address option may not be used for input of the first operand if the first operand is different from the second. The incore address option may never be used for output of the result. The MULTIPLY operation destroys any matrix which has been stored in core using the incore address option (see section III.A - SPECIAL COMMENTS). All calculations are done in double precision. Examples:

```
MULTIPLY (SEQ1)(SEQ2)(SEQ3).
MULTIPLY (SEQ1)(SEQ1)(SEQ2).
MULTIPLY (SEQ2)(I)(SEQ5).
```

OUTPUT (mnemonic: OUT)

The OUTPUT operation outputs a matrix, a row at a time with or without format control, to a user specified data set. If format control is used, the output address specified should not be used anywhere else in the current SOUPAC job step except with options which also perform formatted I/O (e.g. the INPUT and OUTPUT operations of MATRIX). The syntax of OUTPUT is two addresses, input and output addresses respectively, optionally followed by the desired format enclosed by quotation marks. Examples:

```
OUTPUT (SEQ1)(SEQ5) "(20E15.7)".
OUTPUT (SEQ2)(SEQ3).
```

PARTITION (mnemonic: PAR)

The PARTITION operation is used to select a sub-matrix of an original input matrix. The first operand is the input address and the second operand is the output address. The next four parameters specify in order, the beginning column of the partition, the ending column of the partition, the beginning row of the partition, and the ending row of the partition. If either beginning parameter is left out, the partition begins with the first row (or column). If either ending parameter is left, the partition ends with the last row (column). Examples:

```
PARTITION (SEQ5)(SEQ2)(5)(6)(2)(50)
PARTITION (SEQ4)(SEQ2)(3)(40).
```

PERMUTATION (mnemonic: PER)

The PERMUTATION operation permutes, on option, rows or columns or rows and columns of an input matrix. The resulting matrix is output to the second operand output address.

The third operand is an option flag. If the option flag is specified as zero, columns of the matrix are permuted. If the option flag is specified as one, rows of the matrix are permuted. If the option flag is specified as other than zero or one, both rows and columns are permuted.

The order of columns or rows of the output matrix is determined by index sets. Examples:

If the statement

```
PERMUTE (S1)(S2)(0)(5)(1,4).
```

is used and S1 has the matrix

```
 1.   2.   3.   4.   5.
-1.   0.   2.    .1   .3
```

the resulting output matrix will be

```
 5.   1.   2.   3.   4.
 .3  -1.   0.   2.   1.
```

If the input matrix had been

```
 1.    2.    3.    4.    5.    6.
-1.    0.    2.     .1    .3    4.
```

the resulting output matrix would be the same as the output matrix already listed. Note that this implies that only those columns or rows which are explicitly listed will be output.

If the statement

```
PER (S1)(S2)(1)(3)(2)(1).
```

is used and the matrix on S1 is

```
1    2    3
4    5    6
7    8    9
```

the resulting output matrix will be

```
3    2    1
6    5    4
9    8    7
```

If the statement

```
PER (S1)(S2)(2)(3)(2)(1).
```

is used on the matrix

```
1    2    3
4    5    6
7    8    9
```

the resulting output matrix will be

```
9    8    7
6    5    4
3    2    1
```

Warning: When permuting rows alone or rows and columns, it may be necessary to include a prolog card #DEFINE for D49 with appropriate parameters. In addition, space on FT99F001 should be checked. See a SOUPAC consultant for assistance whenever permuting rows.

PRINT (mnemonic: PRI)

The PRINT operation prints out a matrix, one row at a time, under the control of a user supplied format. Formats follow FORTRAN IV conventions with the added restriction that formats are limited to 592 characters.

The first parameter is the address of the matrix to be printed. The second parameter is the format enclosed in quotation marks. (Warning: Allow for carriage control as the first character in output lines. A print line has 133 characters.) Example:

```
PRINT (S2) "(' ',3F20.10)".
```

PUNCH (mnemonic: PUN)

The PUNCH operation has the same syntax as PRINT and is used to punch out a matrix under the control of a user supplied format. (Warning: When punching cards, remember that there is room for only 80 characters per card).

The PUNCH operation always punches two cards in addition to the actual data deck. At the front of the data is punched a DATA format card, and at the end of the data is punched an END# card. Example:

```
PUNCH (S1)"(8F10.2)".
```

RECIPROCAL (mnemonic: REC)

The RECIPROCAL operation has two operands, an input address and an output address. The reciprocal of the elements from the first matrix are used to form an output matrix which is output to the second operand address. Example:

```
RECIPROCAL (S1)(S3).
```

REMAP (mnemonic: REM)

The REMAP instruction is used to change single rows of input into several rows of output or to change several rows of input into single rows of output. The REMAP instruction has three operands; an input address, an output address, and an integer which is to be used as the column dimension of the output address.

Case 1: Map single rows of input into several rows of output. In this case the column dimension of the output address is less than and must divide, the column dimension of the input address matrix. For example, if we have on S1 the data matrix

```
1.    2.    3.    4.    5.    6.
```

and use the instruction

```
REMAP(S1)(S2)(2).
```

the resulting output to S2 would be

```
1.    2.
3.    4.
5.    6.
```

Notice that using the instruction

```
REMAP(S1)(S2)(4).
```

would be an error since 4, the column dimension of the output matrix does not divide 6, the column dimension of the input matrix.

Case 2: Map several rows of input into a single row of output. In this case, the column dimension of the output matrix must be a multiple of the column dimension of the input matrix. Furthermore, if the multiple is some value m, m must divide the number of rows of the input matrix. For example, if we have on S3 the data matrix:

1. 2.
3. 4.
5. 6.

and use the instruction

REMAP(S3)(S4)(6).

the resulting output matrix on S4 would be

1. 2. 3. 4. 5. 6.

Notice that the number of columns of the output matrix is three times larger than the column dimension of the input matrix. Notice also that three divides the number of rows of the input matrix.

REWIND (mnemonic: REW)

The REWIND operation has anywhere from one to twenty-one addresses as parameters. REWIND is used to rewind a sequential file. The REWIND operation needs only to be used with the INPUT operation when it is desired to reread a formatted input file. Example (to input the same formatted file from S3 onto both S1 and S2 under control of different formats):

```
INPUT (S3)(S1)( )(5) "(10X,5F10.0)".
REWIND (S3).
INPUT (S3)(S2)( )(8) "(8F10.0)".
```

ROW DELETE (mnemonic: ROW)

The ROW DELETE operation specifies which rows of an input matrix are to be deleted before sending the result to the output address. The first parameter is the input address. The second parameter is the output address. Rows to be deleted are specified by index sets following the output address. Example:

```
ROW DELETE (I)(S3)(1)(3)(7)(8)(11)(45).
```

RECIPROCAL OF SQUARE ROOT OF DIAGONAL (mnemonic: RSD)

The RSD operation has two operands, an input address and an output address. The reciprocal of the square root of the main diagonal elements from the first matrix are used to form a single row vector which is output to the second operand address. Example:

```
RSD (S1)(S3).
```

SCALAR:(mnemonic: SCA)

The SCALAR operation has three parameters. A floating point number specified by the second operand is multiplied by every element of the matrix specified by the first operand. The result goes to the third operand address.

The second operand may be either a floating point number enclosed in asterisks, or a standard SOUPAC input address. If an address is specified, the first element of the matrix at the address is used for the floating point number. Example:

```
SCALAR(S1) *2.* (S2).
SCALAR(S4)(S3)(S2).
```

SINGLE (mnemonic: SIN)

The SINGLE operation has anywhere from one to twenty-one addresses as parameters. Listing an address as a parameter causes any matrices written on that address to be written in single precision. MATRIX stores all data matrices in double precision unless the user specifies otherwise with the suboperation SINGLE. The listing of an address in a SINGLE statement in one MATRIX program does not carry over in effect to any other MATRIX Program.

```
SINGLE (S1)(S2)(S4).
```

SUBTRACT (mnemonic: SUB)

The SUBTRACT operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of rows and columns as all other input matrices for the use of the operation.

Elements of the second matrix through the next to last matrix are subtracted from corresponding elements of the first matrix. Output goes to the last address. An address may be used more than once as an input address. Examples:

```
SUBTRACT (SEQ1)(SEQ3)(SEQ4).
SUBTRACT (SEQ4)(SEQ2)(SEQ3)(SEQ1)(SEQ5).
```

SUM (mnemonic: SUM)

The SUM operation has three operands; an input address, an output address, and an option indicator.

If option 0 is specified, the resulting output matrix is a single row vector containing the column sum of each column of the input matrix. Specifying no option is equivalent to specifying option 0.

If option 1 is specified, the resulting output matrix is a single column vector containing the row sum of each row of the input matrix.

If the option is specified as any number other than 0 or 1, a single element matrix is output which contains the sum of all elements over the entire matrix. Examples:

```
SUM (SEQ1)(SEQ3).
SUM (SEQ1)(SEQ2)(2).
```

TRANSPOSE (mnemonic: TRA)

The TRANSPOSE operation transposes a matrix (interchanges rows and columns). TRANSPOSE destroys any previous usage of the incore address storage. The two parameters for TRANSPOSE are first the input address and second the output address of the result. Example:

```
TRANSPOSE (SEQ1)(SEQ2).
```

UPPER TRIANGLE (mnemonic: UPP)

The UPPER TRIANGLE instruction copies a matrix from one address to another and sets all elements which are below the main diagonal to zero. It is possible to indicate if it is desired that the main diagonal elements also be set to zero.

The UPPER TRIANGLE instruction has three operands; an input address, an output address, and an integer option flag. If the option flag is omitted or is zero, the main diagonal elements are included as part of the upper triangle. If the option flag is non-zero, the main diagonal elements are set to zero. Examples:

```
UPPER (S1)(S3).
UPPER (S2)(S4)(1).
```

VECTOR (mnemonic: VEC)

The VECTOR operation has two operands, an input address and an output address. A single vector from the first location is used to form a diagonal matrix which is output to the second address. If the input matrix has more rows than columns, the first column vector is used to form the diagonal matrix. If the input matrix has more columns than rows, the first row is used to form the diagonal matrix. Example:

```
VECTOR (SEQ1)(SEQ3).
```

VERTICAL AUGMENT (mnemonic: VER)

The VERTICAL AUGMENT operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of columns as all other input matrices for the use of the operation. An address may be used more than once as an input address.

Input matrices from the first address through the next to the last address are stacked top to bottom and the result goes to the last address. All input matrices must have the same number of columns. Example:

```
VERTICAL AUGMENT (SEQ1)(SEQ2)(SEQ3).
```

III. Special Comments

A. Incore Address Option

Besides the standard SOUPAC addresses, MATRIX also recognizes the additional address I. The I symbol as an address represents internal storage in the machine.

An obvious use of this feature is to cut down on I/O time for matrices which are to be used in future operations within the current matrix program. The internal storage feature also saves time when the user desires his output from an operation to be printed or punched. The user must keep in mind that data cannot be passed to subsequent programs with the I storage. The user should also be aware of the restrictions on I storage as mentioned above in some of the subparameter operations (see INVERT, MULTIPLY, SQUARE, and TRANSPOSE). In all cases the use of this option is not recommended for matrices which do not fit within the memory available to the MATRIX program while running within any particular region size.

1) To add the matrix on SEQ1 to the matrix on SEQ2 leaving the result in core and also printing the result, code as follows:

ADD (SEQ1)(SEQ2)(I/PRINT).

2) To vertically augment the matrices in core, on SEQ1 and on SEQ2, storing the result on SEQ4, code as follows:

VERTICAL AUGMENT (I)(SEQ1)(SEQ2)(SEQ4).

B. Labeled Output

Provided in the MATRIX program is the facility to title and put column labels on any matrix which is printed using normal SOUPAC print conventions. The labeling feature is not allowed with the PRINT matrix operation.

To use the labeling feature, it is first necessary to put the title and labels in a temporary storage area. This is accomplished with the LABEL operation (see Subparameters).

To use a label which has been placed in a temporary storage area, code (L) after the print portion of the output address. If F format is also desired code either (F,L) or (L,F) after the print. Examples:

1) To move (copy) the matrix on SEQ1 onto SEQ5 printing the result in F format with title and column labels, code as follows:

MOVE (SEQ1)(SEQ5/PRINT(F,L)).

2) To add the matrices on SEQ1 and SEQ2 storing the result on SEQ3 and also printing the result with title and column labels, code as follows:

```
ADD (SEQ1)(SEQ2)(SEQ3/PRINT(L)).
```

3) To transpose the matrix stored on SEQ1 to SEQ2, printing out the result in F format with title and column labels, and punching out a card deck of the transposed matrix, code as follows:

```
TRANSPOSE (SEQ1)(SEQ2/PRINT(L,F)/X).
```

TRANSFORMATIONS

I. Purpose

TRANSFORMATIONS is a data manipulations program. Unlike MATRIX, which performs operations on a complete matrix, TRANSFORMATIONS operates upon matrices one row at a time. This strategy provides almost unlimited flexibility in transforming your data. Some of the general uses include creating new variables as functions of present variables, recoding or collapsing data, and reordering or eliminating variables. More advanced uses are facilitated by an instruction set which allows testing and branching depending on single variable characteristics or relations between variables, indirect addressing (FLAG-NOTATION), and inputting and outputting to and from different sequential units during the program.

TRANSFORMATIONS serves several purposes in the SOUPAC system. First, it can be used as a stand alone program to perform computations on your input data and yield the final results. Also, it can be used to prepare your data for input into another SOUPAC program or to make modifications from the output of one program for input into another.

II. Description

The TRANSFORMATIONS program reads in one row of data and executes the program until the end program card or last card instruction appears. It continues to read in data one row at a time, while executing the same program for each successive row until all the rows of data have been processed.

There are 2000 variables allowed in the TRANSFORMATIONS program. Before each row of data is read into the program, variables 1 through 1000 are set to zero. Variables 1001 through 2000 are set to zero only before the initial row of data has been read. Normally, manipulations performed on successive rows of data are independent from each other, but when values are moved to variables over 1000, information can be passed from one row to another or maintained during the processing of the whole matrix. This feature provides for accumulating sums or other totals as well as the capability of having information from previous rows of data determine the kinds or extent of manipulations to be performed on the current row.

Input to the program can be specified by the parameter on the TRANSFORMATIONS card or by the INPUT instruction. Output can only be achieved by use of the OUTPUT instruction.

III. Parameters

The one parameter on the TRANSFORMATIONS card is the input address of the main input matrix. This can be either CARDS or SEQUENTIAL 1-15. This card is followed by the subparameter cards describing the transformations to be performed. The last card must always be an END PROGRAM card.

```
TRA(C).
ADD(1)(2)(3).
DIV(1)(3)(4).
OUT(P) (1,4).
ENDP
```

The main input matrix is from cards and the output is printed.

```
TRA(S3).
LOG(3)(7).
SQU(4)(8).
OUT (S4)(7,8).
ENDP
```

The main input matrix is from SEQ 3 and the output goes to SEQ 4.

IV. Subparameter List

mnemonic	notes	operation name	examples
ABO		abort	ABORT.
ABS	1	absolute value	ABS (1)(51).
ADD	1,2	addition	ADD (1)(2)(52). ADD (1)(7)(11)*8*(53).
ANG	1	angle to radians	ANG (3)(54).
A-C	1,4	arccosine	A-C (4)(55). A-C (5)(56)*0*"BAD".
A-S	1,4	arcsine	A-S (6)(57). A-S (7)(58)(8)"*+1".
A-T	1	arctangent	A-T (9)(59).
C-G		computed go to	C-G (10)"A""B""C""D".
COM	1	combine	COM (11)*10*(12)(60).
CON	3	constant	CON (101)*4.3*. CON (102)(7). CON (103,114)(230,235)*0,5*.
COS	1	cosine	COS (13)(61).
DIF	2	difference if	DIF (1)(5)"X""Y""Z". DIF (6)*9*"N""Z""P".
DIV	1,2,4	division	DIV (14)*10*(62). DIV (15)(6)(63)*0*. DIV (15)(6)(63)*0*"R".
EBC	1	EBCDIC	EBC (16)(64).
EXC	1	exchange	EXC (6)(8).
EXI		exit	EXIT.
EXP	1	exponent base e	EXP (17)(65).
FAC	1	factorial	FAC (18)(66).
FIX	1	fixed point conversion	FIX (19)(67).
FLØ	1	floating point conversion	FLO (20)(68).
GO		go to	GOTO "PLACE".
IF		arithmetic if	IF(1) "*+1""EX""*+1".
INPUT		input from unit	INP (S2)(200). INP (S3)(301)"EOF". INP (S4)(410)"END"(17)"(17F4.0)".
LAS		last card operation	LAST.
ELO	1,4	log base e	ELO (21)(69). ELO (21)(69)*0*"NEG".
LOG	1,4	log base 10	LOG (22)(70). LOG (22)(70)*0*.
MAX	1,2	maximum value	MAX (2)(4)(7)*10*(9)(71).
MIN	1,2	minimum value	MIN (1)(3)(5)(72).

mnemonic	notes	operation name	examples
MOD	1,2	modular arithmetic	MOD (23)*4*(73).
MOV	1	move	MOV (24)(74).
MUL	1,2	multiplication	MUL (1)(25)(26)(75). MUL (27)*3*(76).
NO		no operation	NOOP.
OUT	3	output to unit	OUT (P(F))(1,30). OUT (S3)(1,5)(8)(51,85).
PER	3	permute	PER (701)(6,10)(51,85)(2). PER (801)*1,4*(6)*1,4*(7).
RAD	1	radians to angle	RAD (28)(77).
REC	1,2	recode	REC (29)"GT"(30)(78)*11*. REC (31)"EQ"(32)(79)*0*(79)*1*.
SIG	1	sign transfer	SIG (33)(34).
SIN	1	sine	SIN (35)(80).
SKI	2	skip record on unit	SKI (S2)(100). SKI (S3)*1*.
SQU	1,4	square root	SQU (36)(81). SQU (37)(82)*-1*"IM".
SUB	1,2	subtraction	SUB (38)(39)(83). SUB (40)*2*(84).
SUM	1	summation	SUM (1)(10)(85).
SUP		suppress warnings	SUP.
WAR		warnings on	WARN.
XAD	5	fixed point addition	XADD(41)(42)(49).
XDI	5	fixed point division	XDIV(41)(42)(48).
XIF	5	fixed point arithmetic if	XIF (46)"P""O""T".
XMU	5	fixed point multiplication	XMUL(41)(42)(47).
XSM	5	fixed point summation	XSM (46)(49)(45).
XSU	5	fixed point subtraction	XSUB(41)(42)(46).

NOTES

The following features are available to an instruction if and only if the number appears in the notes for that instruction:

1. DO-notation may be used with variables and floating point constants.
2. Floating point constants may be substituted for input variables.
3. Variable ranges may be specified instead of single variables.
4. Substitute output values and transfer labels may be used to avoid program termination in the case of undefined output values.
5. The input variables must be in fixed point representation.

V. Transformations Labels

In TRANSFORMATIONS there are several instructions which perform some kind of test on your data. In most cases, the results of that test cause the program not to execute the next sequential instruction, but to branch to some other statement in the TRANSFORMATIONS program and continue executing with that statement. In order to refer to these statements we wish to transfer to, TRANSFORMATIONS' labels are employed.

The form of these labels can be illustrated by the following example which skips over the divide statement if the divisor is zero.

```
                IF(3)"NEXT" "JUMP" "NEXT".
"NEXT"          DIV (2)(3)(4).
"JUMP"          next statement
```

or

```
IF(3) "*+1" "*+2" "*+1".
DIV (2)(3)(4).
next statement
```

The preceding equivalent examples exhibit the two types of TRANSFORMATIONS labels. The syntactical rules which govern the two types of labels follow.

Type 1

A. A type 1 label consists of eight or less alphanumeric characters set off by a pair of quotes.

B. Alphanumeric characters consist of the alphabet from A to Z and the numeric digits from 0 to 9.

C. Any unique label may appear as an operand or branch address of any number of TRANSFORMATIONS subparameter instructions.

D. Any label which appears as an operand or branch address of an instruction must appear immediately preceding and be part of at least one and only one TRANSFORMATIONS subparameter instruction in the present TRANSFORMATIONS program.

Type 2

A. A type 2 label consists of a positional reference of the form *+n set off by a pair of quotes.

B. The symbol * is pointing to the statement in which it appears. Therefore, *+1 would point to the next statement, *+2 would skip one statement, and *-1 would point to the preceding statement.

C. This type of label need only appear as an operand or branch address and not before the TRANSFORMATIONS subparameter instruction which it is referencing.

D. Since the statement to which you are going to branch is always indicated relative to the statement from which you are branching, it is possible to point to an address which would lie beyond the end of the program or before the beginning of the program. Needless to say, this would result in an error condition.

E. Branching to "*+0" would create an infinite loop and is also illegal.

Example

If you had a sample with 18 variables and you wanted to eliminate all observations with missing data, you could execute either of the following equivalent programs:

```
        TRA(C).                          TRA(C).
        REC (1,18) "EQ"*-0.*(99)*1*.     REC(1,18) "EQ"*-0.*(99)*1*.
        IF (99) "BAD" "ZERO" "MIS".      IF (99) "*+1" "*+2" "*+3".
"BAD"   ABORT.                           ABORT.
"ZERO"  OUTPUT(S1) (1,18).               OUTPUT(S1) (1,18).
"MISS"  NOOP.                            NOOP.
        ENDPROGRAM                       END PROGRAM
```

The sample is input from cards. The second instruction scans variables 1 through 18 and recodes variable 99 to a 1 if any missing data is found. In this case, we assume that missing data on the data cards has been coded as blanks which are read into the program as minus zeroes. The IF instruction branches to one of the three labels depending if the value of variable 99 is negative, zero, or positive. In this way, if missing data was found, variable 99 will be a 1 instead of a zero and the branch will skip over the OUTPUT instruction causing the observation with missing data to be deleted.

VI. Subparameter Description

ABO

The ABORT instruction causes immediate termination of the TRANSFORMATIO program and the entire SOUPAC program. This instruction is often transferred to when internal tests reveal incorrect data. Example:

ABORT.

ABS

The ABSOLUTE VALUE instruction takes the absolute value of the first variable and stores it into the second variable. Example:

ABS (3)(25).

ADD

The ADD instruction has from three to one hundred parameters pointing to variables. The first variable through the next to last variable are summed and the result is stored into the last variable. Examples:

ADD (6)(7)(23).
ADD (1)(3)(5)(7)*37.4*(100).

ANG

The ANGLE TO RADIANS instruction converts the first variable, which should be a measure of an angle, into radians and stores the result into the second variable. Example:

ANG (3)(17).

A-C

The ARCCOSINE instruction takes the arccosine of the first variable and stores it into the second variable. The first vairable must be between minus one and one inclusive. The result will be stored in radian. Example:

A-C (7)(34).

A-S

The ARCSINE instruction takes the arcsine of the first variable and stores it into the second variable. The first variable must be between minus one and one inclusive. The result will be stored in radians. Example:

A-S (9)(13).

A-T

The ARCTANGENT instruction takes the arctangent of the first variable and stores it into the second variable. The result will be stored in radians. Example:

A-T (8)(70).

C-G

The COMPUTED GO TO has from two to twenty two parameters. The first parameter contains a variable and the following parameters contain labels. The basic form is

C-G(v)"L_1" "L_2" "L_3" ..... "L_n". where $n \leq 21$

The variable must be floating point. If it is not of integral value then it is truncated, (all digits to the right of the decimal point are dropped). The instruction will then branch to the label whose position in the list is equal to the integral value of the variable. Example:

C-G(7)"A" "B" "C" "D" "E".

If variable 7 is equal to 4.0 or 4.3 then the instruction will branch to label "D". If variable 7 is less than 1, then the program will terminate. If it is over 5 the next instruction will be executed.

CON

The CONSTANT instruction contains from two to one hundred parameters. The instruction is used to assign constant values to variables. The first parameter indicates either a variable or range of variables. The subsequent parameters contain the fixed point constant(s) and/or the floating point constant(s) which are assigned to the variable(s). Examples:

CON(7)(99) assigns fixed point value 99 to variable 7.
CON(8)*4.3* assigns floating point value 4.3 to variable 8.
CON(7,8)(99)*4.3* is equivalent to the previous two together.

The more complicated structures which contain ranges and increments for the variables and constants, uses DO-notation which is explained in Sec. 9. An example of the CONSTANT instruction with that structure will be included in that section.

COS

The COSINE instruction takes the cosine of the first variable and stores it into the second. The first variable must be expressed in radians. Example:

COS (1)(17).

DIF

The DIFFERENCE IF instruction contains five parameters. The first two parameters point to variables and the next three contain labels. The second variable is subtracted from the first. If the difference is negative the instruction branches to the first label, if the difference is zero it branches to the second or middle label, and if the difference is positive it branches to the third or last label. Example:

DIF (3)(5) "A" "B" "C".

If variable 3 minus variable 5 is negative the instruction will branch to label "A".

DIV

The DIVIDE instruction contains from three to five parameters. In the case of three parameters, the first variable is divided by the second varia and the result is stored in the third variable. A division by zero will terminate the program. Example:

DIV (1)(2)(30).

In the case of four parameters, the division will take place as normal exce when the second variable is zero. In that event, the fourth variable will be stored into the third variable as a supplied quotient. Example:

DIV (1)(2)(30)(31).

If the fifth parameter is added, it indicates a label to be branched to in the event of a division by zero. The branch will take place after the supplied quotient is stored into the third variable. Example:

DIV (1)(2)(30)(31)"ZERO".

EBC

The EBCDIC instruction converts characters, which are read into the program by means of an A1 format field, into floating point numbers. The first variable contains the character. The result after the table look up will be stored into the second variable. Example:

EBC (6)(89).

The table used for the conversions is located following the subparameter descriptions. This instruction is often used to prepare character codes for input to a SOUPAC FREQUENCY program.

EXC

The EXCHANGE instruction exchanges the contents of two variables. Example:

EXC (1)(2).

EXI

The EXIT instruction causes immediate termination of the TRANSFORMATION program, but will continue to execute the SOUPAC program which follows. Example:

EXIT.

EXP

The EXPONENT BASE e instruction raises e to the power of the first variable and stores the result in the second variable. Example:

```
EXP (31)(704).
```

FAC

The FACTORIAL instruction calculates the factorial of the first variable ani the result is stored in the second variable. Example:

```
FAC(87)(120).
```

FIX

The FIXED POINT CONVERSION instruction converts the floating point variable and the result is stored in the second variable. Example:

```
FIX (2)(9).
```

FLO

The FLOATING POINT CONVERSION instruction converts the fixed point variable indicated by the first parameter and stores the result into the second variable. Example:

```
FLOAT (7)(8).
```

GO

The GO TO instruction unconditionally branches to the label indicated by the only parameter. Example:

```
GO TO "LABEL".
```

IF

The IF statement has four parameters. The first parameter contains a variable and the remaining three parameters are labels. If the variable is negative, zero, or positive the instruction will branch to the first, second, or third label respectively. Example:

```
IF (33) "L1""L2""L3".
```

If variable 33 is positive the instruction will branch to label "L3".

INP

By use of the input instruction you can read in rows of data from sources other than the main input matrix. The first parameter indicates which unit the data will be input from. The second parameter indicates the starting variable number where the row is to be placed. Examples:

```
INPUT(S3)(500).
INPUT(S4)(601)"EØF".
```

The program branches to the label "EOF" if an end of file occurs.

```
INPUT(S5)(701)"B"(12)"(12F4.1)".
```

This form is for formatted input The fourth parameter is the number of variables and the fifth parameter is a standard FORTRAN format.

If S3 contains 64 variables then this input instruction will read the next row from S3 and store it in variables 500 through 563. Be careful to avoid overwriting of existing variables which you need and also of trying to read more or less rows than exist on a particular unit.

LAS

The LAST CARD instruction allows instructions to be performed after the last row of data has been read in and processed. The LAST instruction divides a program into regular and last card segments. The regular section, as is a TRANSFORMATIONS program without the LAST option, is executed once for every row of data. After all the main input data is processed the last card segment is executed once. One of the main uses of the LAST instruction is to analyze data accumulated in variables 1001-2000 during the regular segment. Example:

```
LAST.
```

Only one LAST instruction may be used for TRANSFORMATIONS programs and branching between regular and last card segments is prohibited.

ELO

The LOG BASE e instruction takes the natural log of the first variable and stores it in the second variable. Example:

```
ELOG (3)(17).
```

LOG

The LOG BASE 10 instruction takes the base 10 log of the first variable and stores it into the second variable. Example:

```
LOG (18)(34).
```

MAX

The MAXIMUM VALUE instruction has from three to one hundred parameters pointing to variables. The variable with the largest value from the first variable to the next to last variable is stored into the last variable. Examples:

```
MAX (1)(2)(7).
MAX (1)(7)(8)(11)(15).
```

MIN

The MINIMUM VALUE instruction has from three to one hundred parameters pointing to variables. The variable with the smallest value from the first variable to the next to last variable is stored into the last variable. Examples:

```
MIN (11)(13)(29).
MIN (1)(3)(4)(7)(8)(10)(12).
```

MOD

The MODULAR ARITHMETIC instruction finds the value of the first variable modulus the second variable and stores the result into the third variable. Example:

MOD (1)(4)(7).

MOV

The MOVE instruction stores a copy of the first variable into the second variable. If a value already exists in the second variable it will be overwritten. Example:

MOVE (3)(9).

MUL

The MULTIPLY instruction has from three to one hundred parameters pointing to variables. The first variable through the next to last variable are multiplied together and the result is stored in the last variable. Examples:

MUL (1)(2)(20).
MUL (3)(4)(5)(6)(10).

NO

The NOOP instruction does nothing. Its primary use is when it is preceded by a label and used as a placeholder in the TRANSFORMATIONS subprogram to which many different instructions branch. It is commonly used at the end of a TRANSFORMATIONS subprogram where several isolated groups of instructions all wish to branch to the end. Example:

"END" NOOP.

OUT

The OUTPUT TO UNIT instruction is the only way the TRANSFORMATIONS program can output a row of data. There are two or more parameters. The first indicates the unit to which the row of variables should be output. The other parameters can be either single variables or ranges of variables. Examples:

OUT (S2)(7).

This will output onto S2 a row with one variable.

OUT (S4)(8,14). or OUT (S4)(8)(9,11)(12,14).

This will output onto S4 a row with seven variables, variable eight through variable fourteen. All outputs to the same unit must have the same number of variables output. A more detailed description of types of ranges which are allowed will appear in Sec. 9 on DO notation.

PER

The PERMUTE instruction permutes the order of all or a subset of your variables. It can have from two to one hundred parameters, all indicating variable numbers. The first parameter indicates a starting point of where a string of variables should be placed. The rest of the parameters compose that string of variables. Each of the parameters in that string represent either a single variable or a range of variables. Examples:

PER (100)(3)(10,14)(4,5).

This example places variable 3 in variable 100, variables 10 through 14 in variables 101 through 105, and variables 4 and 5 in variables 106 and 107. A more detailed description of types of ranges which are allowed will appear in Sec. 9 on DO notation ranges.

PER (2)(1,1999).

This example will not propagate variable 1 through all the variables. It will perform the intended purpose of raising all variable numbers up one.

RAD

The RADIANS TO ANGLE instruction converts the first variable, which should be a measure of radians, into an angle and stores the result into the second variable. Example:

RAD (9)(13).

RAI

The RAISE instruction raises the first variable to the power contained in the second variable and stores the result into the third variable. Example:

RAISE (1)(2)(7).

REC

The RECODE instruction recodes variables depending on the satisfaction of a set of conditions. The sequence of parameters depends on the number of conditions that must be met. The RECODE instruction introduces a new set of terminology which follows:

A. RELATIONAL OPERATORS

"LT"	less than
"LE"	less than or equal to
"EQ"	equal
"NE"	not equal
"GE"	greater than or equal
"GT"	greater than

B. CONNECTIVES

"AND"
"OR"

C. CONDITION SET

A condition set consists of three parameters, the first and third parameters pointing to variables and the second parameter containing a relational operator. Any condition set is either true or false depending upon whether the two variables satisfy the conditions of the relational operator.

D. RECODE SET

A recode set consists of two parameters. The first parameter indicates a variable. The second parameter indicates a variable or a floating point constant. If a recode set is executed the value of the second variable or floating point constant is stored into the first variable.

The RECODE instruction consists of from one to twenty one condition sets joined together in the case of more than one by connectives. This is followed by a recode set to be executed if the logical product of the condition sets is true and optionally a second recode set to be executed if the logical product is false. Examples:

```
REC (4) "EQ" *3* (4) *1* (4) *0*.
```

If variable 4 equals 3.0 then recode it to 1.0, if not then recode it to 0.0.

```
REC (6) "GE" (10) "AND" (7) "LE" (20)(110) (111)
```

If variable 6 is greater than or equal to variable 10 and variable 7 is less than or equal to variable 20 then recode variable 110 to the value of variable 111.

SIG

The SIGN instruction places the sign of the first variable on the second variable. It is often used for saving signs of variables during intermediate calculations. Example:

```
SIGN (1)(901).
```

SIN

The SINE instruction takes the sine of the first variable and stores it into the second variable. Example:

```
SIN (1)(11).
```

SKI

The SKIP instruction has two parameters. The first parameter is a sequential unit. The second is a variable or floating point number. The number in the variable or the floating point number indicate the number of rows to be skipped on the specified unit. Example:

```
SKIP (S2)(7)
SKIP (S5) *1*.
```

SQU

The SQUARE ROOT instruction takes the square root of the first variable and stores it into the second variable. Example:

SQU (13)(14).

SUB

The SUBTRACT instruction subtracts the second variable from the first variable and stores the result into the third variable. Example:

SUB (7)(9)(10).

SUM

The SUM instruction sums a string of consecutive variables starting with the variable indicated by the first parameter and ending with the variable indicated by the second parameter and places this sum into the variable pointed to by the third parameter.

COM

The COMBINE instruction has four parameters. They must be a variable, a floating point constant, and two variables in that order. The first variable is multiplied by the floating point constant and the variable in the third parameter is added to the product. The result is placed in the variable pointed to by the fourth parameter.

SUP

The SUPPRESS instruction stops the printing of all warning messages.

WAR

The WARNING instruction causes all warning messages to be printed. This is the normal condition unless a SUPPRESS instruction is used.

XAD - fixed point addition	XAD (1)(2)(20).
XDI - fixed point divide	XDI (3)(4)(21).
XIF - fixed point if	XIF (5)"T""O""P".
XMU - fixed point multiply	XMU (6)(7)(22).
XSM - fixed point sum	XSM (8)(15)(23).
XSU - fixed point subtract	XSU (16)(17)(24).

The preceding fixed point instructions have the same parameters as the corresponding floating point instructions, except that the arithmetic instructions are restricted to two input variables and division by zero in the XDIVIDE instruction will terminate the program. The variables used in the fixed point instructions must be in fixed point representation rather than the normal floating point representation.

ZAP

The ZAP instruction zeros out variables 1001 through 2000.

VII. EBCDIC Conversion Table

Character	Floating Point Number	Character	Floating Point Number
blank	-0.	¢	74.
0	0.	.	75.
1	1.	<	76.
2	2.	(	77.
3	3.	+	78.
4	4.	|	79.
5	5.	&	80.
6	6.	!	90.
7	7.	$	91.
8	8.	*	92.
9	9.	)	93.
A	10.	;	94.
B	11.	¬	95.
C	12.	-	96.
D	13.	/	97.
E	14.	,	107.
F	15.	%	108.
G	16.	_	109
H	17.	>	110.
I	18.	?	111.
J	19.	:	122.
K	20.	#	123.
L	21.	@	124.
M	22.	'	125.
N	23.	=	126.
O	24.	"	127.
P	25.		
Q	26.		
R	27.		
S	28.		
T	29.		
U	30.		
V	31.		
W	32.		
X	33.		
Y	34.		
Z	35.		

VIII. DO-Notation

DO-notation is a facility provided in the TRANSFORMATION program to enable a user to easily and compactly perform an operation on a set of variables instead of performing that operation on each of the variables individually. This concept of DO-notation corresponds to the concept of FORTRAN DO-loops. The following form of DO-notation would be used in a parameter which points to a variable.

(V_1, V_2, I)

V_1 = the initial variable of the set

V_2 = the criterion variable for termination of the set.

I = the increment

Examples:

(1, 5, 1) points to variables 1, 2, 3, 4, and 5.

(1, 5, 2) points to variables 1, 3, and 5.

(4, 14,3) points to variables 4, 7, 10, and 13.

If the increment is not specified it is assumed to be 1, Example:

(2, 10, 1) is equivalent to (2, 10).

A. The major use of DO-notation is to indicate repetition of an instruction on different sets of variables. Examples:

ADD(1,5,2)(6,8)(12,14)	is equivalent to	ADD(1)(6)(12). ADD(3)(7)(13). ADD(5)(8)(14).
MUL(1,4)(100)(101,107,2)	is equivalent to	MUL(1)(100)(101). MUL(2)(100)(103). MUL(3)(100)(105). MUL(4)(100)(107).

If, in an instruction containing several parameters using DO notation, the sets of variables are unequal in length, the instruction will cycle until the longest set of variables has been satisfied. The variables used in the shorter sets after they have been exhausted will be the last variables of that set. Example:

MAX(1,7,3)(14,15)(13,15,2).	is equivalent to	MAX(1)(14)(13). MAX(4)(15)(15). MAX(7)(15)(15).

B. The secondary use of DO-notation, called DO-notation ranges, is used with the constant, output, and permute instructions. Instead of indicating a repetition of the instruction on each variable in the set, the instruction is executed once. The parameter in which the

DO-notation range occurs now points to a string of variables. Examples:

OUTPUT(S1)(2,7). outputs the string of variables 2,3,4,5,6 and 7.

OUTPUT(S2)(3,7,2). outputs the string of variables 3,5, and 7.

PERMUTE (100)(3)(10,15)(20,24,2).
has two parameters of DO-notation ranges in one instruction. The instruction places into variables 100 through 109, the following variables: 3, 10,11,12,13,14,15, 20,22,24

Looking at the CONSTANT instruction, we see DO-notation ranges used to indicate strings of fixed and floating point constants as well as strings of variables.

The first parameter of CONSTANT can point to a variable or a string of variables. The later is the only case which involves DO-notation ranges, so the discussion will be confined to that case.

CON(2,4)(1)(2)(3). places the fixed point constants one, two and three, into the string of variables 2,3,4

By using DO-notation ranges with fixed point constants the equivalent instruction would be

CON(2,4)(1,3).

The next two examples are also equivalent. They both assign to variables five through eight the floating point constants 2.5,5.0,7.5, and 10.0.

CON(5,8)*2.5**5.0**7.5**10.0*.
CON(5,8)*2.5,10.0,2.5*.
The initial values of the string of floating point constants is 2.5. The termination criterion is 10.0. The increment is 2.5

All three types of DO-notation ranges can be used together in the CONSTANT instruction. The following example combines both of the preceding sets of examples into one instruction.

CON(2,8)(1,3)*2.5,10.0,2.5*.

Note: If, when using DO-notation ranges with the CONSTANT instruction, the string of variables is unequal in length to the total number of fixed and/or floating point constants indicated, the string of constants is truncated if it is longer and if it is shorter the last constant is assigned to the remainder of the variables.

IX. Flag Notation

Flag notation is TRANSFORMATION's version of indirect addressing. Instead of a parameter pointing directly to a variable, it points indirectly to a variable through another variable. The parameter points to a variable which in turn points to another variable. This feature enables an instruction to point to different sets of variables depending upon the values assigned to the intermediate variables.

The main type of flag notation is called F-flag notation. F-flag notation is indicated by inserting an F directly after the variable number. Example:

```
ADD(7F)(8F)(9F).
```

Restriction: The values in the intermediate variables of flag-notation (variables 7,8, and 9 in this example) must be in fixed point representation and must point to a valid variable number.

Let the notation Vn indicate variable n. Example: V7 indicates variable 7.

```
If    V7 = 100
      V8 = 150
      V9 = 180
```

then the preceding example would generate after the indirect addressing takes place:

```
ADD(100)(150)(180).
```

F-flag notation can also be used with DO-notation. Example:

```
If    V100 = 2    V110 = 6    V120 = 10
      V101 = 13   V111 = 16   V122 = 19
      V102 = 5    V112 = 9    V124 = 13
```

then MUL(100F,102F)(110F,112F)(120F,124F,2). would generate after indirect addressing.

```
MUL(2)(6)(10).
MUL(13)(16)(19).
MUL(5)(9)(13).
```

Note that the limits of the DO-notation are extracted form the intermediate variables and not from the final variables.

The other type of flag notation is called D-Flag notation. Its only use is with DO-notation. The difference between it and F-flag notation is that the limits of the DO-notation with D-flag notation are derived from the final variables after indirect addressing takes place. Example:

Using the same variable values as above.

```
MUL(100D,102D)(110D,112D)(120D,124D).
```

would generate after indirect addressing:

```
MUL(2,5)(6,9)(10,13).

        or

MUL(2)(6)(10).
MUL(3)(7)(11).
MUL(4)(8)(12).
MUL(5)(9)(13).
```

X. A TRANSFORMATIONS Example

Let us consider a set of data consisting of ten variables. Variable one contains a zero for females and a one for males. Variables two through five and eight through ten all range from zero to ninety-nine and are to be collapsed into the numbers one to four representing quartiles. Variables six and seven are to be recoded into a dichotomous variable depending if the value is five or not five. The output is then split into males and females suitable for input into two separate FREQUENCY programs. Also desired are averages from variables two through ten before recoding takes place. Totals are kept during the regular segment and then in the last card segment these are divided by the sample size and printed.

```
       TRANSFORMATIONS(C).                          inputs one row from cards
       PERMUTE (6)(8,10)(6,7).                      reorders the variables
       ADD (1001)*1*(1001).                         increments row number by one
       ADD (1002,1010)(2,10)(1002,1010).            adds respective values to row totals
       RECODE (2,8)"LT"*25*(2,8)*1*.                recodes first quartile values
       RECODE (2,8)"GE"*75*(2,8)*4*.                recodes fourth quartile values
       RECODE (2,8)"GE"*50*(2,8)*3*(2,8)*2*.        recodes third & second quartile values
       RECODE (9,10)"EQ"*5*(9,10)*1*(9,10)*0*.      creates dichotomous variables
       IF (1)"BAD""FEM""MALE".                      branches based on male or female
"BAD"  ABORT.                                       aborts program due to bad data
"FEM"  OUTPUT(S2)(2,10).                            outputs female data
       GO TO "END".                                 branches around male output
"MALE" OUTPUT(S3)(2,10).                            outputs male data
"END"  NOOP.                                        this instruction is only a placeholder
       LAST.                                        indicates beginning of last card segment
       DIVIDE (1002,1010)(1001)(1002,1010).         calculates averages
       OUTPUT (P)(1001,1010).                       prints sample size and averages
       END PROGRAM
```

The regular segment is executed once for every row of card input data. Here the values are recoded and output, while also row totals are kept in variables greater than 1000. The last card segment is executed once to calculate and print the averages.

XI. Notes and Ideas

1. Missing data of the form -0.0 can be differentiated from 0.0 only in the recode statement, so if this distinction must be made, -0.0 should first be recoded to another value before testing.

2. The valid outputs in the OUTPUT instruction are PRINT and/or Sn, $n \leq 15$.

3. When collapsing data be careful not to overlap your recoding and inadvertently recode values twice or more.

4. For those not familiar with the terminology in the RECODE instruction more examples appear below.

Given:	V1 = 10	V3 = 13	V5 = 23
	V2 = 23	V4 = 89	V6 = -7

The following condition sets have the respective truth values.

(1) "GT" (2) is false
(3) "LE" (4) is true
(2) "EQ" (5) is true
(6) "GE" *0* is false
(2) "NE" (5) is false
(6) "LT" (4) is true

If two or more condition sets are joined by "AND", they must all be true for the logical product to be true. If two or more condition sets are joined by "OR", then the logical product is true if any of the condition sets are true. If the connectives are mixed, then the "AND" connective is of higher precedence than the other connectives in the same way that multiplication is of higher precedence than addition in ordinary arithmetic.

5. The A-C, A-S, ELOG, LOG and SQU instructions have optionally available third and fourth parameters, a variable and a branch address. In the case where the output is not defined or the input variable is invalid, the third parameter is substituted for the output value and branches to the branch address. If the branch address is not specified, processing continues with the next instruction.

Examples:
```
SQU(1)(18)(200).
SQU(1)(18)(200)"ERR".
```

6. Do not use F flag notation with the CONSTANT, PERMUTE, or OUTPUT instructions.

7. If you are inputting from a double precision matrix and have over 500 variables, beware! Change the input matrix to single precision before TRANSFORMATIONS or see Bill Walter in the SOUPAC Office.

BASIC POPULATION STATISTICS PACKAGE

BASIC POPULATION STATISTICS

In this section are the programs commonly used to provide the user with a "first look" at his data. It is neither expected, nor is it a good idea, that the researcher be naive about his data (it is, of course, assumed that any hypothesis to be tested was made previous to collecting the data). Tabulations, or "frequency counts," cross tabulations, sample means, rank scores, etc. are all useful "summary statistics" that may indicate warning signals concerning assumptions made by the experimenter that might be questioned.

In addition, the statistics required for many basic techniques will be found in this section. These include, for instance, the sample mean, and other statistics derived from the moments of a sample, rank order statistics, the non-parametric Mann Whitney U statistic, and the χ^2 statistic for testing independence of variables (appropriate for considering the hypothesis of independence of two or more variables from the same sample).

FREQUENCY COUNTING AND MEASURES OF ASSOCIATION

I. General Description

This program computes tables of the frequency of occurrence of values that input variables take, and where appropriate, measures of association may be computed. Input to the program may be in the form of previously computed tables (on which measures of association will be computed) or may be in the form of raw data. Only integer numbers may be counted; decimal point data will be rounded. Negative values are allowed.

A. FREQUENCY COUNTING

The following options are available:

1. Either one-dimensional or two-dimensional tables may be specified. For one-dimensional counts, the frequency of occurrence for each value of the variable is listed. For two-dimensional counts, each value of the second variable is counted separately for each value of the first variable.

2. Control variables may be used which enable counting to be done in up to 12 dimensions. If control variables are specified, data must be presorted on these variables. When the value of any variable designated a control variable changes from one row to the next, counting is stopped and a new table is started. Thus counting proceeds as long as the values of all control variables remain constant.

3. The minimum and maximum values to be attained may be specified separately for each variable to be counted (or, optionally, not specified at all). If either the maximum or minimum values are not specified, they will be determined from the data using an extra read of the data. Values which fall below the minimum or above the maximum are ignored. This capability adds flexibility to the program and may be an appreciable cost saver. Its misuse by gross estimates of minimum and maximum values can, however, be costly.

4. For each cell in a one-dimensional table, the percentage, if requested, of the total sample that were counted there will be printed. For two dimensional tables, the percentage of the row and column may also be requested.

5. A weighting variable may be specified. Without a weighting variable, frequency counts are advanced by one for each occurrence of a value. When a weighting variable is used, the frequency counts are advanced by the value of the weighting variable for the row. Thus some rows of data may be given more importance than others.

6. Labels can be given for variables so that output is more readable. Each label is restricted to eight characters or less.

7. Input may be from previously computed two-dimensional tables, from which measures of association can be directly computed.

B. MEASURES OF ASSOCIATION

The following coefficients are calculated and printed on option for two-way tables:

1. Chi-square and related coefficients

Let: n = total population of the table

n_{ab} = number of Vertical classification a (column a) and Horizontal classification b (row b)

$n_{a.} = \sum_b n_{ab}$

$n_{.b} = \sum_a n_{ab}$

α = number of rows
β = number of columns

Then: chi-square = $\sum\sum_{ab} \dfrac{\left(n_{ab} - \dfrac{n_{a.}\, n_{.b}}{n}\right)^2}{n_{a.}\, n_{.b}/n}$

adjusted chi-square (Yate's correction for continuity) for 2 x 2 tables only =

$$\sum\sum_{ab} \frac{\left(n_{ab} - n_a\, n_b - 1/2\right)^2}{\dfrac{n_a\, n_b}{n}}$$

$$C = \left(\frac{\text{chi-square}/n}{1 + \text{chi-square}/n}\right)^{1/2}$$

$$T = \left(\frac{\text{chi-square}/n}{(\alpha-1)(\beta-1)}\right)^{1/2}$$

C and T are measures of contingency and can be looked up in contingency tables. The maximum expected frequency is also printed.

2. Lambda coefficients

Let: $n_{am} = \underset{b}{\text{Max}}\ n_{ab}$

$n_{mb} = \underset{a}{\text{Max}}\ n_{ab}$

$n_{.m} = \underset{b}{\text{Max}}\ n_{.b}$

$n_{m.} = \underset{a}{\text{Max}}\ n_{a.}$

$$\text{Lambda} = \frac{\sum_a n_{am} + \sum_b n_{mb} - n_{.m} - n_{m.}}{2n - n_{.m} - n_{m.}}$$

$$\text{Lambda H} = \frac{\sum_a n_{am} - n_{.m}}{n - n_{m.}}$$

$$\text{Lambda V} = \frac{\sum_b n_{mb} - n_{m.}}{n - n_{m.}}$$

Lambda coefficients will be indeterminate if all values lie in one column or row.

Lambda H can be defined as the decrease in probability of error in predicting the H-variable when knowledge of the value of the V-variable is considered as opposed to random guessing of the H-variable.

Ninety-five per cent confidence limits are calculated and printed for Lambda H and Lambda V using the methods discussed by Goodman and Kruskal in their second article. (See references). Lambda is always between Lambda H and Lambda V.

3. Weighted Lambda Coefficients

$$\text{Weighted Lambda H} = \frac{\sum_a \frac{n_{am}}{n_{a.}} - \underset{b}{\text{Max}} \sum_b \frac{n_{ab}}{n_{a.}}}{\alpha - \underset{b}{\text{Max}} \sum_a \frac{n_{ab}}{n_{a.}}}$$

$$\text{Weighted Lambda V} = \frac{\sum_b \frac{n_{mb}}{n_{.b}} - \underset{a}{\text{Max}} \sum_b \frac{n_{ab}}{n_{.b}}}{\beta - \underset{a}{\text{Max}} \sum_b \frac{n_{ab}}{n_{.b}}}$$

These are Lambda H and Lambda V calculated using weighted quantities $1/\alpha \frac{n_{ab}}{n_{a.}}$ and $1/\beta \frac{n_{ab}}{n_{.b}}$, respectively, instead of n_{ab}.

No confidence limits are provided for the weighted lambda coefficients.

4. Gamma Coefficient

$$\text{Let: } PS = \sum_{ab} n_{ab} \left[\sum_{a'>a} \sum_{b'>b} n_{a'b'} \right]$$

$$PD = \sum_{ab} n_{ab} \left[\sum_{a'>a} \sum_{b'<b} n_{a'b'} \right]$$

$$\text{Then Gamma} = \frac{PS - PD}{PS + PD}$$

Ninety-five per cent confidence limits for Gamma are calculated using the method outlined and preferred in the second article by Goodman and Kruskal.

C. References

These coefficients are discussed and compared by Leo A. Goodman and William H. Kruskal in their article "Measures of Association for Cross Classification", American Statistical Association Journal, December, 1954.

The Gamma coefficient is their suggested measure.

The C coefficient was first suggested by Karl Pearson and the T coefficient is due to Tchuprow.

The Lambda coefficients apparently were first suggested by Louis Guttman ("The Predication of Personal Adjustment", Bulletin 48, Social Science Research Council, New York, 1941).

The development of the approximate sampling theory and of the machinery for calculating the confidence intervals for Lambda and Gamma was done in a sequel article by Goodman and Kruskal: "Measures of Association for Cross Classification III; Approximate Sampling Theory", American Statistical Association Journal, June, 1963.

The statistics that are requested will be printed immediately following each table.

I. Restrictions

The program is limited currently to 450 input variables and 1000 tables. Tables are restricted to a maximum of 80,000 cells, each of which can hold a maximum count of 32,767. As many tables as will fit into work storage (80,000) will be computed in each read of the data. If tables will fit into 80,000 cells, card input is allowed. If maxima and minima are not specified for card input, data will be transferred to disk during preread of data.

III. Parameters

A. Main Parameter Card

Immediately following the program name FREQUENCY (mnemonic: FRE), the following parameters are listed, each enclosed in parentheses with a period after the last parameter used:

Parameter Number	Use or Meaning
1	Input Address.
2	0 - ignore blanks 1 - count blanks separately 2 - count blanks as zeroes
3	Spacing: 0 - normal spacing 1 - one table per page
4	Address of labels.
5	Variable number of weight variable.
6	Type of input: 0 - raw data n - where n is the number of previously computed tables. If n > 1, then input must be from cards, and each table is a separate data deck.

If both parameters 1 and 4 are cards, the labels must precede data.

B. Subparameters

Subparameters follow the main parameter card and can be in any order. A period must follow each subparameter statement though the statement can be continued on more than one card. If the subparameter statement is left out, the option is not used. In the following explanation I = integer and F = real number.

Mnemonic	Use or Meaning
PER(I)(I)(I).	Per cents are requested: 0 - no, 1 - yes 1st integer = total per cent 2nd integer = row per cent 3rd integer = column per cent
MIN*F**F*.,..... . MAX*F**F*..... .	Minimum and maximum are given. The last value is propagated to any remaining variables. Data will be reread if either MIN or MAX is missing.

Mnemonic	Use or Meaning
MEA(I)(I)(I)(I). Only applicable to 2-way tables	Measures are requested 0 - no 1 - yes 1st integer = X^2 (with a code of 2 both X^2 and a table of expected frequencies will be printed) 2nd integer = λ (lambda) 3rd integer = weighted λ 4th integer = γ (gamma)
CONTROL(I)(I)..... .	Up to 10 control variables are allowed. The I's should be the variable numbers of the control variables.
ONE(I,I,I)(I,I,I)..... . TWO(I,I,I)(I,I,I)..... .	One and only one of these two must be in every program. ONE means one-way tables. TWO means two-way tables. In ONE, (I,I,I) specifies one range of tables. In TWO, (I,I,I)(I,I,I) specifies one range of 2-dimensional tables.

The notation (I,I,I) has the following meaning: If it is absent completely, i.e., ONE. or TWO. then all possible tables are calculated. The first integer is the initial value, the second is the terminal value and the third is the increment. It means: take all values starting at the first integer and stepping by the third integer until you reach the second integer. If the third integer is missing, the increment is taken to be one. If the second is also missing, then the first is taken as a single table specification. As many as wanted can be specified subject to the following restrictions: In the two-way tables, no more than 540 separate ranges, i.e., (I,I,I)(I,I,I) can be specified.

IV. Labels

Labels can come from cards or temporary storage. Each label should be treated as if it were two variables each 4 characters long. For example, if there are 6 variables in the input data, then there would be twelve variables for labels and the data card would be DATA(12)(12A4).

All the labels are treated as one row of input n variables long. Labels need not be given for each variable but if a variable is skipped and more labels follow, then it should be replaced with eight blanks.

V. Examples

1.
```
FRE(C).
PER(1).
CONTROL(1)(3).
ONE(2,6,2)(9).
END P
```

Input is from cards; per cent of totals will be printed; control variables are 1 and 3; resulting tables are 2, 4, 6, and 9. Blanks will be ignored.

2. FRE(C)(2)(1)(C)(2).
   PER(1)(1)(1).
   TWO(3)(4)(2,6,2)(1,5,2).
   END P

All per cents will be given; 2 will be the weighting variable; resulting tables will be 3 vs 4; 2 vs 1; 2 vs 3; 2 vs 5; 4 vs 1; 4 vs 3; 4 vs 5; 6 vs 1; 6 vs 3; 6 vs 5. Tables will be printed one per page and blanks will be counted as zeroes.

Since both labels and data are on cards the deck will look like this:

```
FRE(C).............   .
:
:
:
END S
DATA(n)(nA4)
label for first labeled variable.....label for last labeled variable
END#
DATA(n/2)(.....)
:
:
:
END#
```

3. FRE(S1)(1).
   TWO.
   MEA(2)(1)(1)(1).
   END P

All possible two-way tables will be calculated; all four measures will be calculated and the table of expected frequencies will be printed. Blanks will be counted separately.

4. FRE(C)()()()()(2).
   MEA(1)()(1).
   END P

Input is in the form of two previously computed tables. X^2 and weighted λ will be calculated.

Since there are two tables the deck will look like this:

```
FRE(C).............   .
:
:
:
END S
DATA...............
:
:
:
END#
DATA...............
:
:
:
END#
```

VI. Output Examples

A. ONE-DIMENSIONAL TABLE

A one-dimensional frequency table might be output as follows:

VALUE	1	4	5	9	TOTAL
FREQ	3	25	30	2	60

This table indicates that the value 1 occurred 3 times, that the value 4 occurred 25 times, and so on, for a total of 60.

B. TWO-DIMENSIONAL TABLE

A two-dimensional frequency table might look like this:

VARIABLE 1 ACROSS
VARIABLE 2 DOWN

VALUE	1	3	5
2	1	4	3
5	2	8	1
7	1	2	1

SAMPLE SIZE = 23

This table would indicate that simultaneous observations of 1 for variable 1 and 2 for variable 2 occurred once. A value of 3 for variable 1 at the same time as a value of 5 for variable 2 occurs 8 times. The number of observations in the sample was 23.

RANK ORDERING PROGRAM

I. General Description

A. Purpose

The RANK ORDERING program receives as input raw data matrix and produces as output a matrix in which each element has been replaced by a number denoting the rank of the element WITHIN ITS COLUMN. In other words, each column of the input matrix is considered a separate variate and will be converted to a corresponding ranking.

The smallest variate-value is assigned rank 1.0, the next largest a rank 2.0, etc., until the largest variate-value is assigned the highest rank. In the case of tied values, identical ranks are assigned to equal values, the rank-number being set equal to the average of the rank which would occur if the tied values were distinguishable. This is sometimes known as "mid-rank method".

B. References

Kendall, Maurice G., Rank Correlation Methods, Charles Griffin and Co., Ltd. London, 1948.

II. Restrictions

A. Input

The input data to this program may come from any source. If cards are used as input, the number of rows in the input matrix must be specified on the data format card and the total number of elements in the matrix may not exceed 30,000. The maximum number of rows for any matrix input to this program is 30,000, and the maximum number of columns for any matrix input to this program is 450.

B. Output

If an input matrix contains more than 30,000 elements, an automatic partitioning of the input data occurs such that each partition contains the maximum number of complete columns possible within the constraint that no one partition may contain more than 30,000 elements.

The results of the ranking of each partition are output separately, one partition per output address specified as a parameter on the program parameter card. A maximum of twenty-one such output address are allowed.

CAUTION: If partitioning is anticipated, the user should specify one output address for each partition anticipated. This warning applies especially in the case where printed or punched output occurs. Printing and punching will occur only for the partitions for which it is specified.

The exception is for partitions over the twenty-first one. For partitions beyond the twenty-first, printing and punching is done if it was specified for the twenty-first partition. However, no partitions beyond the twenty-first one may be stored on a peripheral device (SEQUENTIAL address).

C. Data

Since all comparisons in this program are done in single word length operands, in some cases the program may not be able to successfully differentiate between two values which agree through the first five significant digits and differ in subsequent digits.

III. Parameters

The parameters for the RANK ORDERING program must follow the program name on the program call card in the order given below:

Parameter Number	Use or Meaning
1	Input Address.*
2-23	Output Address.

IV. Special Comments

If RANK ORDER correlation coefficient ρ (Spearman's rho) is desired, the rankings should be input to the CORRELATION program (see individual program description) and the Product Moment Correlation coefficient obtained.

*If CARDS are used the DATA card must contain the number of rows as well as the number of columns in the input matrix (See User's Guide for detail.

STANDARD SCORES

Note: Sample Size, Mean, Standard Deviation, Variance, Skewness and Kurtosis are referred to as "Basic Statistics" in this write-up.

I. General Description

This program is used to calculate the following:

Mean: $\bar{X}_j = \frac{\sum_{i=1}^{N} X_{ij}}{N}$ Variance: $V_j = \frac{N \sum_{i=1}^{N} X_{ij}^2 - \left(\sum_{i=1}^{N} X_{ij}\right)^2}{N(d.f.)}$

where degrees of freedom:

d.f. = N or N-1 see parameter 4 for explanation

Standard Deviation: $S_j = \sqrt{V_j}$

Skewness: $\frac{\sum_{i=1}^{N} X_{ij}^3 - \bar{X}_j\left(3.0 * \sum_{i=1}^{N} X_{ij}^2 - \bar{X}_j\left(3.0 * \sum_{i=1}^{N} X_{ij} - N * \bar{X}_j\right)\right)}{NS_j^3}$

Kurtosis: $\frac{\sum_{i=1}^{N} X_{ij}^4 - \bar{X}_j\left(4.0 * \sum_{i=1}^{N} X_{ij}^3 - \bar{X}_j\left(6.0 * \sum_{i=1}^{N} X_{ij}^2 - \bar{X}_j\left(4.0 * \sum_{i=1}^{N} X_{ij} - N * \bar{X}_j\right)\right)\right)}{NS_j^4} - 3.0$

Standardized Scores: $Z_{ij} = \frac{X_{ij} - \bar{X}_j}{S_j}$

Standardized Scores about Mean = A, and Std. Dev. = B: $Z_{ij} * B + A$

Moving Averages: $\bar{X}_j = \frac{\sum_{k=j}^{b+(j-1)} X_k}{b}$ where b = length of the period

II. Restrictions

A. The maximum number of variables is 450.

B. "Basic Statistics" may be calculated using as many as 30 control variables. Data must be presorted (for instance with SORT-MERGE or on a card sorting machine) on the control variables.

C. In obtaining moving averages where nvar = number of variables and b = length of period, nvar*b is fixed for any core size. If a design will not fit, a message will be printed giving proper increment for Region.

D. Moving averages are exclusive of all other options.

E. Output is of four categories:
   1. With or Without Control Breaks
      a. "Basic Statistics"
      b. Moving averages: "Basic Statistics"
   2. Without Control Breaks
      a. Standard scores about data mean and standard deviation. Printed output includes "Basic Statistics."
      b. Standard scores about a given mean and standard deviation. Printed output includes "Basic Statistics."
      c. Both Sections 2a and 2b at the same time and output may be printed and/or stored on two different storage locations.
      d. Control breaks may not be used with Standard Scores option.

III. Parameters

The parameters appear on the program call card following the program name STANDARD SCORES (or the program mnemonic STA) in this order:

Parameter Number	Description
1	Input Address. SEQUENTIAL 1-15. Cards if only "Basic Statistics" desired, or if precalculated means and standard deviations are supplied (see parameter 10).
2	Output Address of Standardized Scores.
3	Output Address for "Basic Statistics." "Basic Statistics" can be put out on a temporary unit. Output is in the form of six column vectors (N, $\bar{X}$, S_j, V_j, $Skew_j$, $Kurt_j$).
4	If 1, use N-1, if 0, use N for denominator of standard deviations. N-1 gives an unbiased estimate of the population standard deviation and population variance. N gives the sample standard deviation and sample variance.
5	Output Address for Standard Scores about a specified Mean and Standard Deviation. SEQUENTIAL 1-15 and/or PRINT.
6	If parameter 5 is being used, place desired Mean between asterisks, for example, *50*.

Parameter Number	Description
7	If parameter 5 is being used, place desired Standard Deviation between asterisks, for example, *5*.
8	Moving Averages: Put the number of periods (observations) over which it is desired that the data be averaged (i.e. b). If control variables are being used and/or the actual number of observations is less than stated, the data will be averaged using the actual number of observations.
9	If set equal to 1, "Basic Statistics" will be corrected for missing data. Missing data must be coded as -0.0 (blanks).
10	Input Address of Means and Standard Deviations. First row contains means, second row contains standard deviations. All additional rows input are ignored. Valid only for standard scores.

If using controls, on a separate card immediately after the STANDARD SCORES card, list variable numbers of those variables used as controls. For example, if controlling on variables 1, 2, and 4:

```
STA(S2)(P).
$C-B(L)(2)(4).
```

NOTE: If there is only 1 observation and parameter 4 is set to 1, then the "Basic Statistics" for that variable will be set to zero. If blanks are checked and standard scores are requested, those observations which have a blank will remain blank after calculation of standard scores.

IV. Example 1

(Use of Parameter 10)

```
//  EXEC SOUP
//SYSIN  DD  *
MAT.                                    1
MOV(C)(S1).
END P
COR(S1)(S2).                            2
MAT.                                    3
TRA(S2)(S3).
END P
STA(S1)(P)()()()****()()(S3).           4
END S
DATA(10)(10F1.0)
  .
  .
  .
END#
/*
```

Example 2

```
//  EXEC SOUP
//SYSIN  DD  *
MAT.                                    1
MOVE(C)(S1/P).
END P
STA(S1)()(P).                           2
STA(S1()(P)(1).                         3
STA(S1)(P)(P).                          4
STA(S1)()()()()****(5).                 5
END S
DATA(36)(36F2.0)
  .
  .    Data deck is placed here
  .
END#
/*
```

Explanation of Example 1

Program 1 will move data from cards to Sequential 1.

Program 2 will calculate means, standard deviations, and sample size, and save results in column form on Sequential 2.

Program 3 transposes contents of Sequential 2 and stores them in row form on Sequential 3.

Program 4 reads means, standard deviations, from Sequential 3 (ignoring all subsequent rows) and calculates and prints Standard Scores.

Explanation of Example 2

Program 2 will print "Basic Statistics" for variable$_j$ based on sample standard deviation.

Program 3 will print "Basic Statistics" for variable$_j$ based on unbiased estimate of population standard deviation.

Program 4 will print standard scores matrix in addition to "Basic Statistics."

Program 5 will print "Basic Statistics" over every set of 5 periods (moving averages).

ANALYSIS OF VARIANCE PACKAGE

BALANOVA 5

Table of Contents

Chapter 1. General Description

1.1 Introduction

BALANOVA 5 is a general analysis of variance program applicable to a wide-range of balanced designs. In the case of designs with a replication factor, BALANOVA 5 allows inequality in the number of replications in each cell. If the number of replications is equal or proportional, the analysis is handled by least squares (weighted means). If the number of replications is not proportional then an unweighted means analysis is performed. This is an approximation to the least squares solution.

BALANOVA 5 accepts some designs that are not completely crossed, namely those nested designs in which all main factors are balanced. Hence hierarchical designs are allowed. As well, repeated measures designs are allowed. In these designs the replication factor is not nested in all the other factors.

The design model may be fixed-effects, random-effects or mixed. BALANOVA 5 automatically determines all the legal sources of variation (main effects and interactions) and determines the correct denominator mean square for those sources which can be tested by F test. In order to do this, BALANOVA 5 first generates the expected mean square table which is printed in readable form. The method used closely follows Scheffé (1959), Chapter 8.

BALANOVA 5 will accept most of the designs described in Winer (1962), Chapters 3, 4, 5, 6, and 7 and Lindquist (1953), Chapters 3, 5, 6, 7, 8, 9, 10, and 13 (Types I, III, VI). Chapter 2 and Appendix A of this write-up contain a large number of examples drawn from these two books.

For proper use of BALANOVA 5, the following general warnings should be kept in mind:

1. A general program such as BALANOVA 5 encourages the use of statistics in a "cook-book" manner. Data is generated to fit the input specifications of the program with no consideration given to the theory of analysis of variance. The experimenter who uses a computer program in this way often neglects to consider whether the statistical test is appropriate for the work he is interested in and whether the assumptions needed for the test are satisfied in the particular experiment he has used.

2. Results printed by a program such as BALANOVA are often accepted by the experimenter as being infallible when, in fact, all calculations on a digital computer are subject to possible programming error, round-off error or simple machine error. Always double check your results to make sure they "make sense."

3. In the particular case of analysis of variance, the idea has become widespread that the summary table of F ratios is the most important part of the analysis. This is not the case. The most important part of analysis of variance

is the estimation of the main effects and the interactions. Only by looking at their size can the experimenter evaluate what is happening in his experiment. In order to encourage this use of analysis of variance, BALANOVA 5 prints a table of marginal means which allows easy calculation of all the effects in the experiment. The F table is only a set of warning signals. A non-significant F indicates that the corresponding differences between effects can be attributed to sheer chance.

4. BALANOVA 5 performs an unweighted-means analysis when the replication numbers are non-proportional. The author fears that this option will be used too often and without consideration of its dangers. The unweighted-means solution is often not satisfactory and references on analysis of variance should be consulted. (Scheffé, 1959, Winer, 1962, Lindquist, 1953).

BALANOVA 5 was designed to reduce the great amount of hand computation needed in analysis of variance calculations. It was not intended to eliminate the necessity of the user being familiar with the theory of analysis of variance. It is hoped that the above comments will discourage some indiscriminant use of BALANOVA 5.

1.2 Special Features

The output from BALANOVA 5 consists of

1. A table of the expected mean squares in readable form.

2. The number of replications in each cell in the case of designs with a replication factor.

3. The table of marginal means. All means entering in the computation of the sum of squares are printed.

4. The analysis of variance summary table including, for each source of variation, the sum of squares and mean square, and for each source with denominator, the F ratio and the probability of the chance occurance of the F ratio.

A feature of BALANOVA 5 is its flexible specification of analysis of variance designs, allowing a wide range of designs to be described by a common code.

A large number of checks are made by BALANOVA 5 to ensure that the design is legal and that the data correspond to the design. Diagnostics are printed to indicate all error conditions.

1.3 Legal Design of BALANOVA 5

Consider the following definitions, taken from Scheffé (1959). Let there be p factors in a design, not counting the replication factor, if there is one. A cell is specified by a set of p levels, one for each factor. The layout of design is complete if there is at least one observation in every cell. The factors in such a design are completely crossed. If the design is complete and there is a replication factor (i.e. all cells have at least one observation and at least one cell has more than one observation) then the design is considered to be a Class A design in BALANOVA 5.

There are many analysis of variance designs which are not complete in the above sense. Examples of incomplete designs are Latin-square, incomplete blocks and nested designs. The only incomplete designs which are allowed in BALANOVA 5 are nested designs which are balanced in all factors except for the replication factor (which need not be balanced). These incomplete designs are called Class B and C designs. "Nesting", "balanced" and "replication factor" are defined in the next three paragraphs. These definitions are illustrated in Chapter 2.

Nesting may be defined as follows: The levels of a factor C are nested within the levels of a factor A (in short, C is nested within A) if and only if each level of C appears with only a single level of A in the observations. Note that if C is not nested within A, it is crossed with A, but only if every level of C appears with every level of A is C completely crossed with A. Latin-square and incomplete block designs are only partly crossed.

A nested factor C is balanced if the number of levels of C is the same within each combination of those factors within which C is nested and the factors (if any) which are crossed with C are completely crossed.

A replication factor, in BALANOVA 5, is a factor which is nested within one or more other factors, but not necessarily within all other factors. Furthermore, no factor may be nested within the replication factor. That is, a factor is a replication factor if and only if for every other factor A in the design, it is either nested within A or crossed with A. A replication factor may be nested within some factors and crossed with others. There can be at most one replication factor in a design.

The distinction is made between replication factors and other nested factors in BALANOVA 5 since replication factors do not have to be balanced. All other factors must be balanced.

Using these definitions, the following designs are legal in BALANOVA 5.

Class A designs (completely crossed with nested replications)

Class A designs contain (p + 1) factors of which p are the main factors and the other factor is the replication factor. The following two conditions must both be met for the design to be Class A.

(a) All p main factors are completely crossed.

(b) The replication factor is nested in all main factors.

Thus one-way and factorial designs are Class A designs.

Class B designs (other replication designs)

Class B designs also contain (p + 1) factors of which p are the main factors and the other factor is the replication factor. However one or both of the two conditions, (a) and (b), are not satisfied in Class B designs.

When (a) is not satisfied, that is, the p main factors are not completely crossed, then the main factors must satisfy the following condition.

(a') Consider any two main factors, A and B. Either A is completely crossed with B, or A is nested within B or B is nested within A. This must be true for all pairs of main factors. Furthermore, at least one pair must have the nested relationship or else (a) would be satisfied.

When (b) is not satisfied, then the following condition must be true.

(b') The replication factor is nested in at least one but not all main factors. Note that the requirement that the replication factor be nested in at least one factor is part of the basic definition of a replication factor.

Class B designs then can be of the following two types.

Hierarchical designs: (a') and (b) are satisfied. The replication factor is nested in all factors but there is some nesting among the main factors.

Repeated measures designs: (b') is satisfied. Either (a) or (a') can be satisfied. The necessary feature (b') of repeated measures designs is that the replication factor is crossed with one or more of the main factors. The factors in which the replication factor is nested may themselves be either crossed (a) or nested (a').

Class C designs (no replication factor)

Class C designs have p factors and there is no replication factor. All factors must be balanced. For each pair of factors, e.g. factors A and B, either A is completely crossed with B, or A is nested within B or B is nested within A. There does not necessarily have to be any nesting at all.

In summary, then, designs are classed in the following way in BALANOVA 5. Class A and B designs have a replication factor, Class C designs do not. Class A designs are distinquished from Class B designs in that a Class A must have 1) all main factors completely crossed and 2) the replication factor nested in all main factors. Class B designs violate one or both these requirements.

In Class A and B designs, the replication factor does not need to be balanced. However all nested factors, except the replication factor (if any), must be balanced. Recall that in Class A designs, the replication factor is the only nested factor.

As explained above, the replication factor is distinquished from other nested factors since it does not have to be balanced. There are two other reasons for distinguishing the replication factor from other nested factors. These reasons are important even if the replication factor is balanced.

1. In Class A designs (completely crossed with replications) only cell means are stored in the computer and thus very large designs can be accommodated. The allowable number of replications in each cell is virtually unlimited.

2. For all replication designs, whether of Class A or B, the level number for the replication factor in each nest does not need to run from one up to the maximum number of levels in each nest as it does for all other factors. Any convenient numbering of the replications may be used (e.g. a unique number for every subject in the experiment, regardless of the nest within which he is). This feature of BALANOVA 5 is especially useful when several dependent variables are analyzed and there is missing data for some of the subjects for some of the dependent variables.

1.4 Calculations for equal and unequal number of replications

The calculations performed by BALANOVA 5 for designs with a replication factor (Class A and B designs) depend on whether the number of replications in each cell are equal or unequal. If the numbers are equal, the standard analysis of variance calculation is made (least-squares or weighted means analysis). If the numbers are unequal, a check is first made to see if the cell N's are proportional. In a two-way analysis of variance, for example, the cell N's are proportional if the number of replications in the ij cell, N_{ij}, satisfies

$$N_{ij} = \frac{N_{iT} \times N_{Tj}}{N_{TT}}$$

where the T's indicate marginal totals. If the cell N's are proportional, BALANOVA 5 makes the least-squares calculations, i.e. weighted means are used. If the cell N's are not proportional, then the method of unweighted means is used (See Scheffé, pp. 262-3 or Winer, pp. 222-4).

In general, if i, j, k,..., λ are those factors within which the replication factor is nested (not necessarily all the factors in the design), and if $N_{ijk...\lambda}$ is the number of replications in a particular nested cell, then the cell N's are proportional, if, for all combinations ijk...λ,

$$N_{ijk...\lambda} = \frac{N_{iTT...T} \times N_{TjT...T} \times \cdots \times N_{TTT...\lambda}}{(N_{TTT...T})^{q-1}}$$

In this formula, the T's indicate marginal totals and Q is the number of factors within which the replication factor is nested. In particular, the one-way analysis with unequal N's is a proportional design (i.e. the cell N's are proportional) by this definition, since

$$N_i = \frac{N_i}{(N_T)^0} = \frac{N_i}{1} = N_i$$

In fact, any design in which the replication factor is nested in only one factor is a proportional design.

1.5 Parameters

Each observation (row of data) input to this program must be identified by a number for each factor including the replication factor. These numbers (which cannot be read in I format) represent the levels of the corresponding factors and must precede the dependent variables. In the output produced by the program, each factor is given a unique letter name, beginning with A. Thus the first column of the input data corresponds to the levels of factor A which is described on the first factor specification card (see below). Each additional factor is given the next letter in the alphabet, and a corresponding factor specification card. The dependent variables follow the factor levels on the input data, and they are numbered one through the total number of dependent variables, in the output of the program.

On the program call card, the following parameters follow the program name, BALANOVA 5; with the first five parameters being required.

Parameter Number	Description
1	Input Address. CARDS or SEQUENTIAL 1-15.
2	Number of factors counting replication factor if there is one. Maximum = 10.
3	Number of dependent variables.
4	Number of levels of the 1st factor.
5-13	Number of levels of the 2nd-10th factors.

Parameter Number	Description
14	1 if desire unweighted means analysis even though have proportional cell frequencies.
15	1 to suppress printing of all means.

Following the program card is a separate subparameter card (factor specification card) for each factor in the order in which the factors appear in the input data. Each card has the following parameters.

Parameter Number	Description
1	0 if fixed factor 1 if random factor
2	0 if not the replication factor 1 if is the replication factor
3-11	Factors in which this factor is nested

As in other SOUPAC programs, parameters at the end of the card which are not used may be deleted and the period appear after the last non-zero parameter. The factor specification cards must be followed by an END PROGRAM card.

1.6 Specification of a Design

Any design is described by listing the following information about each factor in the design, including the replication factor if there is one. The information for each factor is punched on a separate card (a factor specification card), and the cards should be in the same order as the factors are in the input data. Each parameter should be enclosed in parentheses, and each card terminated by a period.

Parameter 1	Type of factor. The first parameter on each factor specification card should be a zero if the factor is fixed, and a one if it is random. The replication factor is <u>always</u> a random factor. At least one factor in every design must be random.
Parameter 2	<u>Replication Factor</u>. If the design has a replication factor, this is indicated by punching a one for the second parameter. A design may have only one replication factor. If there is no replication factor, the second parameter should be zero (or blank) on all of the factor specification cards.
Parameter 3-11	<u>Nesting</u>. The factors in which the given factor (the one to which this card refers) is nested are listed. Factors are numbered from one through the number of factors in the design.

If the factor is not nested, parameters 3-11 may be completely omitted.

An example of this way of specifying a design will be now given. Consider a two-way analysis of variance with subjects within cells. The design is considered to have three (not two) factors, namely A and B, the main factors, and C, the replication factor. Suppose there are 3 levels of A and 4 levels of B and that each cell has 10 subjects. The cards used to perform this analysis are listed below. Each line corresponds to one IBM card.

```
BALANOVA(CARDS)(3)(1)(3)(4)(10).
(0)(0).
(0)(0).
(1)(1)(1)(2).
END PROGRAM
```

The first card listed above calls the BALANOVA program. The first parameter is the location of the data (CARDS), the second is the number of factors (3), the third is the number of dependent variables (1), the fourth is the number of levels of the first factor (3), followed by the number of levels of the second factor (4), and finally the number of levels of the last factor, which is the maximum cell size (10) when considering the replication factor.

The second card is the factor specification card for factor 1. The first 2 parameters are zero, labeling this factor as fixed, and as not being the replication factor. The third card is the factor specification card for factor 2 which is also fixed, and not the replication factor. Note that parameters 3-11 are blank, as factors 1 and 2 are not nested in any other factors.

The fourth card is the factor specification card for factor 3. Its four parameters denote it as a random factor, as the replication factor, and as nested in factors 1 and 2. The fifth card terminates the BALANOVA program.

Chapter 2. Design Examples

2.1 Class A Designs: (Completely crossed with nested replications)

Single-factor designs (Winer, Chapter 3; Lindquist, Chapter 3)

The single-factor design is the simplest analysis of variance design. It is often called the one-way analysis of variance or the simple-randomized groups design. See the Winer and Lindquist references and also Hays, Chapters 12 and 13, for a detailed discussion of this design.

The program cards for a design with 5 groups of 20 subjects each is shown below. The groups are considered to be levels of factor 1 and the subjects are factor 2. The order of the factors is the same on the main parameter card, on the factor cards, and punched on the input cards.

```
BALANOVA5 (C)(2)(1)(5)(20).
(0).
(1)(1)(1).
END P
```

Note that in this design no subject appears in more than one group. Hence the subject factor (factor 2) is nested within the group factor (factor 1). Note also that the replication factor is listed as a random factor while the group factor is fixed. It would be possible to consider the group factor as a random factor. See, e.g., Winer, pp. 56-63. However, in the single-factor design the calculations are unchanged regardless of what type the main factor is. So the choice of fixed or random for the type of the main factor is immaterial in the input to Balanova 5. The interpretation of the results, however, depends on the type of factor assumed.

If the groups are of unequal size, the program does not have to be altered except that the number of levels of the replication factor must be greater than or equal to the number of replications in the largest groups. For example, if the groups have sizes 10, 12, 15, 9, 20, then the above program is still correct.

Factorial designs (Winer, Chapter 5 and 6; Lindquist, Chapters 5, 8, 9, 10)

In these designs, subjects are assigned to groups. Each group is identified by a set of levels, one level for each main factor in the design. The levels of a factor may represent different experimental treatments or a classification of a continuum into discrete levels.

Three examples from Winer and Lindquist are given below. Further examples of Class A designs may be found in Appendix A.

Winer, pp. 233-238

This is a 2 x 3 factorial design with three observations per cell. The program cards are:

```
BALANOVA5 (C)(3)(1)(2)(3)(3).
(0).
(0).
(1)(1)(1)(2).
END P
```

Note that the subject factor (factor 3) is nested in both factor 1 and factor 2, since any subject is only treated by one level of factor 1 and one level of factor 2. The replication factor is random while the main factors are fixed. Either or both of the main factors could be random. The F tests would vary depending on this choice. Balanova 5 makes the correct tests depending on the factor type indications. See Winer, pp. 170-174.

Winer, pp. 241-244

This is a 2 x 4 factorial design with unequal cell frequencies. The factors are specified in the same way as for an equal cell frequency design except that the number of levels of the replication factor is set greater than or equal to the maximum cell frequency. The calculations are carried out using the method of unweighted means unless the cell frequencies are proportional, in which case the least squares analysis of variance calculations are carried out.

```
BALANOVA5 (C)(3)(1)(2)(4)(5).
(0).
(0).
(1)(1)(1)(2).
END P
```

As mentioned in Winer (p. 243, below Table 6.3-3) both main factors are assumed to be fixed.

Lindquist, pp. 226-228

This is a completely crossed design with proportional cell frequencies. The program cards are:

```
BALANOVA5 (C)(4)(1)(2)(4)(3)(8).
(0).
(0).
(0).
(1)(1)(1)(2)(3).
END P
```

No special indication is needed that the design is proportional. Balanova 5 will discover that, for all ijk,

$$N_{ijk} = \frac{N_{iTT} N_{TjT} N_{TTk}}{(N_{TTT})^2}$$

where N_{ijk} is the number of subjects in cell ijk and a T in a subscript indicates a marginal total. For example, cell 122 (the 5th column of the chart at the bottom of page 226) has $N_{ijk} = 6$. Now

$N_{1TT} = 4 + 4 + 4 + 6 + 6 + 6 + 8 + 8 + 8 + 5 + 5 + 5 = 69$

$N_{T2T} = 6 + 6 + 6 + 6 + 6 + 6 = 36$

$N_{TT2} = 4 + 6 + 8 + 5 + 4 + 6 + 8 + 5 = 46$

N_{TTT} = total number of replications = 138

$$\frac{N_{1TT}N_{T2T}N_{TT2}}{(N_{TTT})^2} = \frac{69 \times 36 \times 46}{(138)^2} = 6 = N_{ijk}$$

A similar equality will be found for all ijk. Hence the design is proportional.

2.2 Class B designs (other replication designs)

All Class B designs have a replication factor but at least one of the two conditions for a Class A design is not satisfied. In repeated measures (below), the replication factor is not nested in all the other factors in the design. In hierarchical designs (below), the non-replication factors are nested rather than crossed.

Repeated measures designs (Winer, Chapter 7; Lindquist, Chapter 13)

In repeated measures designs, the replication factor is not nested within all the other factors as it is in Class A designs. The replication factor is crossed with one or more factors in the design.

Four examples from Winer are given below. Lindquist's Type I, IV and VI are similar to these designs and are listed in Appendix A. The designs discussed in Winer, Chapter 4, (Single-factor experiments having repeated measures) are not considered to be Class B designs since the subject factor is not nested in any factor and hence cannot be considered to be a replication factor in Balanova 5. Such designs are considered to be Class C designs.

All repeated measures designs require additional assumptions to ensure the validity of the tests. It is suffested that Scheffé and Winer be consulted about these assumptions.

Winer, pp. 302-318

This design has two main factors 1 and 2 with subjects nested within factor 1. Factor 1 is the group factor. The program cards for the data in Table 7.2-3 are:

```
BALANOVA5 (C)(3)(1)(2)(4)(3).
(0).
(0).
(1)(1)(1).
ENDP
```

Note that factor 3 (subjects) is a random factor and that it is nested only in factor 1. Also note that although there are six subjects altogether, there

are only 3 within each nest, namely within each level of factor 1 and hence the number of levels for factor 3 (subjects) is 3. As will be pointed out in more detail in Chapter 3, the subjects may retain their original numbering from 1 to 6 rather than being numbered from 1 to 3 in each nest as would be the case if factor 3 (subjects) were not a replication factor.

As discussed in Winer, p. 318, either or both of factors 1 and 2 may be random rather than fixed, in which case Balanova 5 will change the tests appropriately and print out the appropriate expected mean square table.

Winer, pp. 319-337

This is a repeated measures design with repeated measures on two completely crossed factors. The program cards for the data in Table 7.4-3 are:

```
BALANOVA5 (C)(4)(1)(2)(3)(3)(3).
(0).
(0).
(0).
(1)(1)(1).
END P
```

Note that factor 4 (subjects) is only nested within factor 1 while factors 1, 2 and 3 are completely crossed and factor 4 (subjects) is crossed with factors 2 and 3. Factor 4 (subjects), again, is a random factor.

On page 335, Winer discussed the tests when some of the factors 1, 2 or 3 are random. Balanova 5 follows these rules automatically.

Winer, pp. 337-349

The designs in this section also have three main factors but the replication factor is nested in two of them. The program cards are:

```
BALANOVA5 (C)(4)(1)(2)(2)(4)(3).
(0).
(0).
(0).
(1)(1)(1)(2).
END P
```

Again the general form of the expected mean squares in the case of some of the random factors is given on pp. 347-348. Balanova 5 prints out the expected me square table appropriate to the design chosen and makes the correct tests.

Winer, pp. 374-378

The case of unequal group size is also handled by Balanova 5. If the number of replications within each nest is proportional, then the exact analysis of variance is performed. Winer calls this the least-squares solution in Table 7.8-6. Note that if the replication factor is nested in only one factor then it is proportional and the least-squares solution will be performed.

If the number of replications is not proportional, then an unweighted-means analysis is performed.

If it is desired in the proportional case to perform the unweighted-means analysis anyway, an override is available. See main program card, parameter 14 in Section 1.5.

The factor specification table for the unequal number of replications case is identical to the equal case except that the entry under Number of Levels must be the maximum number of levels of the replication factor in any one nest. For example, the data in Table 7.8-3 would have the program cards:

```
BALANOVA5 (C)(3)(1)(2)(3)(5).
(0).
(0).
(1)(1)(1).
END P
```

For these data, the design is proportional, and Balanova 5 would normally perform the least squares solution. The unweighted solution could be performed instead if an override is given. Both solutions (see Tables 7.8-5 and 7.8-6) have been checked with Balanova 5.

Hierarchical designs (Winer, Chapter 5; Lindquist, Chapter 7)

In hierarchical designs, the replication factor is nested in all the main factors, but all the main factors are not crossed. Some of the main factors are nested. Two designs from Winer are illustrated below. Further examples from Winer and Lindquist are given in Appendix A.

Winer, pp. 184-187

The hospitals within drugs example on p. 184 has the following program cards if there are n = 20 patients in each hospital.

```
BALANOVA5 (C)(3)(1)(2)(3)(20).
(0).
(0)(0)(1).
(1)(1)(1)(2).
END P
```

Hospitals (factor 2) are nested within drugs (factor 1) since each hospital appears with only one drug. The number of levels for hospitals is three rather than six since there are only three hospitals in each level of drug. Patients are nested in hospitals and drugs and patients (factor 3) is the replication factor since no factor is nested within patients.

Note that the factorial design (Class A) given on the bottom of p. 185 has program cards:

```
BALANOVA5 (C)(3)(1)(2)(6)(10).
(0).
(0).
(1)(1)(1)(2).
END P
```

Hospitals (factor 2) is no longer nested in drugs (factor 1) and the number of levels of hospitals (factor 3) is now 6 rather than 3. Furthermore, there are only 10 patients in each cell.

The partially hierarchical design on p. 186 (Table 5.12-1) has factor specification table (for n = 20):

```
BALANOVA5 (C)(4)(1)(2)(3)(2)(20).
(0).
(0)(0)(1).
(0).
(1)(1)(1)(2)(3).
END P
```

Factors 1, 2 and 3 are called factors A, B and C in Winer.

2.3 Class C designs (no replication factor)

Class C designs have no replication factor and hence must be balanced. The factors may be crossed or nested. Several examples are given below and more are given in Appendix A.

Winer, pp. 111-116

The designs in Winer, Chapter 4, are repeated measures designs but the subject factor is not nested. Therefore the subject factor cannot be considered a replication factor and the design cannot be Class B. The design on pp. 111-116 has the following program cards:

```
BALANOVA5 (C)(2)(1)(4)(5).
(0).
(1).
END P
```

Note that the person factor 2 is of random type. At least one factor in any design must be random for there to be a denominator term for an F ratio. Factors 1 and 2 are completely crossed; therefore, no nesting is indicated. Also there is no replication factor, so this parameter is left blank. Balanova 5 will make the correct test in this design, namely testing the drug factor, factor 1, against the interaction mean square.

The rationale for the test is in Winer, pp. 116-124. The test of homogeneity of covariance is not made by Balanova 5 just as no test of homogeneity of variance is made for any design input to Balanova 5.

Winer, p. 289

The Aborn et al design has no replication factor and hence is a Class C design. The analysis, after the transformation of the data and estimation of missing data, involves pooling the interactions into one mean square and using the pooled estimate as the denominator term to test the three main effects. This procedure cannot be done automatically in Balanova 5. The following steps are suggested. Use the following program:

```
BALANOVA5 (C)(3)(1)(6)(4)(3).
(0).
(0).
(1).
END P
```

Note that one of the factors is stated to be random. This is done solely in order to allow Balanova 5 to function since the program requires that there be a legal denominator term. If all factors were fixed, there would be no denominator. The summary table printed by Balanova 5 will be used only for the sum of squares, not for the F ratios, which will be incorrect since they are based on a model having a random factor and in the real design no factor is random. Pool the interaction sum of squares, form the mean squares and carry out the correct F tests by hand.

The preliminary square root transformation of the data and estimation of missing data must be done by a Transformations program.

Nested designs of Class C

The impression may have been given that no nest factors are allowed in Class C designs. This is not the case. Class C designs have no replication factor (which would be a nested factor) but other factors can be nested besides replication factors. Consider the following design:

```
BALANOVA5 (C)(3)(1)(3)(3)(10).
(0).
(0)(0)(1).
(1).
END P
```

A concrete realization is perhaps hard to give but this design means simply that factor 2 is nested in factor 1 and both factors are crossed with factor 3 (subjects).

Chapter 3. Preparation of Input

3.1 Introduction

The following rules apply to the assignment of factor levels in all types of designs. There is usually no need to rekeypunch existing data however, as it is almost always possible to create the factor levels in the TRANSFORMATIONS program. If you need help using this program, see a SOUPAC consultant.

(a) Non-replication Factors in Class A, B and C designs

The levels for non-replication factors must run from one (1) consecutively up to the number of levels given on the BALANOVA call card. E.g., if a factor represents four treatment groups, these groups must be numbered 1, 2, 3, and 4 and each subject's row or rows in the data matrix must have a 1, 2, 3, or 4 punched to indicate the group he is in. If the factor is nested, the level numbers must run from one (1) up, in each cell of the nest. See the example given in Section 3.2.

(b) Replication Factors in Class A design

The replication numbers (level numbers) can be anything, for example, a subject identification number. The subject numbers do not have to be unique either in a group or between groups. In fact, to tell the truth, in Class A designs, the replication level is not used but it must nevertheless appear, even if it is a dummy. This statement does not apply to other design classes.

(c) Replication Factors in Class B designs, of repeated measures type

Special care must be taken with the replication levels in those designs Let us divide the non-replication factors into two groups:

α-set: those factors in which the replication factor is nested.

β-set: all other factors - i.e. those factors crossed with the replication factor.

If the β-set is empty, the design is of hierarchical rather than repeated measures type. See paragraph (d) below.

Let us denote by an α-cell a particular set of levels of the factors in the α-set. The replications in this cell may be any values (not necessarily from one (1) up) but must be distinct. Again, the numberings in two differe α-cells do not have to be distinct, but can be. In other words, if the replication factor is subjects, an identification number may be used as the replication level. Now each subject appears in more than one row (card) of the data matrix since each subject appears with every combination of levels of the factors in the β-set. Now it should be obvious that every row that refers to the same subject has to have the same replication number. This is the only way that BALANOVA 5 can tell that two different rows refer to the same subject.

(d) Replication Factors in Class B designs, of hierarchical type

The subject numbers must all be different within any one cell (one level set of the non-replication factors) but may be the same over different cells.

Special note on missing data

If a dependent variable field on a card is totally blank, BALANOVA 5 does not include the score in the analysis for the given dependent variable. However other non-blank dependent variable fields on the same card will be included in their respective analyses.

Do not confuse this deletion of missing data with an error comment by BALANOVA 5 to the effect that there is no data cell A = 1, B = 2. This comment means that no data card with A = 1, B = 2 had non-blank data for the given dependent variable.

3.2 Data matrix examples

Class A design

In all the following examples, it is assumed data is stored on sequential file number 1 (SEQUENTIAL 1).

Consider a two-way design with three subjects in each cell. For the purpose of BALANOVA, subjects are also considered to be a factor, the replication factor. Suppose that there are two dependent variables, and further that the factor specification cards are listed in the order given below, following the main program card.

```
BALANOVA(SEQUENTIAL 1`(3)(2)(2)(3)(3).
(0)(0).
(1)(1`(1)(3`.
(0)(0).
END PROGRAM
```

Note that, contrary to the usual case, the replication is the second factor. This illustrates one flexible feature of BALANOVA 5. A data matrix could be:

1	1	1	20	19
1	2	1	8	8
1	3	1	4	4
1	4	2	-3	6
1	5	2	4	10
1	6	2	2	3
1	7	3	4	4
1	8	3	6	2
1	9	3	8	4
2	10	1	2	7
2	11	1	-4	8

2	12	1	25	2
2	13	2	126	15
2	14	2	2	20
2	15	2	3	3
2	16	3	4	4
2	17	3	5	-1
2	18	3	3	2

Note that the first column is the A level, the third column is the C level and the second column is the replication level, which in Class A designs can be anything. The last two columns are the dependent variables. Each row of the data matrix would be punched on one or more cards. A possible format would be (3F5.0,3X,2F6.0).

The order of the rows is immaterial. They could be in any order and have been written in a systematic order only for convenience.

Class B design - repeated measures

The data in Winer, Table 7.2-3 could be analyzed with the following program, using three factors and one dependent variable.

```
BALANOVA(SEQUENTIAL 1)(3)(1)(2)(4)(3).
(0)(0).
(0)(0).
(1)(1)(1).
END PROGRAM
```

Data Cards:

1	1	1	0
1	2	1	0
1	3	1	5
1	4	1	3
1	1	2	3
1	2	2	1
1	3	2	5
1	4	2	4
etc.			
2	1	5	5
2	2	5	4
2	3	5	6
2	4	5	6
2	1	6	7
2	2	6	5
2	3	6	8
2	4	6	9

Again, the rows could be any order. The subjects in the second level of facto
A could be assigned level numbers 1, 2, 3 or any other thre distinct numbers.
A possible format for this matrix is (2F5.0,F6.0,1X,F7.0).

Another repeated measure design is Winer, Table 7.4-3. Here we have four factors and one dependent variable.

```
BALANOVA(SEQUENTIAL 1)(4)(1)(2)(2)(3)(4).
(0)(0).
(0)(0).
(1)(1)(1)(2).
(0)(0).
END PROGRAM
```

Data Cards:

```
1    1    1    1    18
1    1    1    2    14
1    1    1    3    12
1    1    1    4     6
1    1    2    1    19
1    1    2    2    12
1    1    2    3     8
1    1    2    4     4
etc.
1    2    6    1    18
1    2    6    2    10
1    2    6    3     5
1    2    6    4     1
2    1    7    1    16
2    1    7    2    10
2    1    7    3     8
2    1    7    4     4
etc.
2    2   12    1    16
2    2   12    2    12
2    2   12    3     8
2    2   12    4     8
```

The assignment of levels to factors A, B, D (columns 1, 2, 4) has to be as shown above, but the subjects numbers can be changed provided that, for each (A,B) cell, no two subjects have the same number. A possible format is (2F5.0, 1X,2F5.0,F12.0).

Class B design - hierarchical

Consider the following design with three factors and one dependent variable.

```
BALANOVA(SEQUENTIAL1)(3)(1)(2)(2)(4).
(0)(0).
(0)(0)(1).
(1)(1)(1)(2).
END PROGRAM
```

This is a hospitals (factor 2) within drugs (factor 1) design, illustrated below, with unequal cell size.

Data Cards:

```
1    1    1    5.0
1    1    2    4.2
1    2    1    5.6
1    2    2    3.2
1    2    3    4.6
2    1    1    5.3
2    1    2    8.2
2    1    3    4.3
2    1    4    6.3
2    2    1    5.7
2    2    2    6.8
```

Note that the hospitals are numbered 1, 2 in each level of factor 1 even though there are four different hospitals involved. This is necessary since factor 2 is a non-replication factor - see the rules about Data Cards in Section 3.1. There are 2 patients in hospital 1 for drug 1, 3 patients in hospital 2 for drug 1, 4 patients in hospital 1 for drug 2 and 2 patients in hospital 2 for drug 2. The patient numbering is flexible - it could be a different number for every patient, regardless of hospital. A possible format is (3F5.0,F7.1).

Class C design

The example in Winer, Table 4.3-1, could be set up as follows, with two factors and one dependent variable.

```
BALANOVA(SEQUENTIAL 1)(2)(1)(5)(4).
(1)(0).
(0)(0).
END PROGRAM
```

Data Cards:

```
1    1      30
1    3      16
2    1      14
2    2      18
2    3      10
2    4      22
1    2      28
3    1      24
3    2      20
3    3      18
3    4      30
4    1      38
1    4      34
4    2      34
4    3      20
4    4      44
5    4      30
5    3      14
5    2      28
5    1      26
```

The rows have been written in a non-systematic order to emphasis that, without exception, in BALANOVA 5 the data rows can be in any order. A possible format is (2F5.0,F10.0).

Chapter 4. Program Details

4.1 Method and program flow

The program follows the procedures in Scheffé (1959), Chapter 8. Scheffé's discussion will not be repeated here, but only a general description of the program flow will be given. The names of the subroutines used are indicated in case reference is made to the program listing. Many of the minor steps and subroutines are not described.

1. Main Program

Calls subroutines and routes your data through the program.

2. Design input and check (INPUTD)

The Factor Specification Cards are read and checked for errors. Many of the error conditions mentioned in Section 4.3 are checked in INPUTD. The design is transformed into the symbolic notation of live, dead and absent subscripts as in Scheffé.

3. Derivation of all legal sources (LEGALS,NEWS)

All possible interactions are generated but only one interaction with a given set of subscripts is retained. The procedure is identical to Scheffé, p. 277, para. 1. The program now has a list of all legal sources (including the original factors).

4. Expected mean squares (AUXIL,EMS)

The expected mean squares for each source are, of course, not computable numbers, but rather symbolic expressions. (cf. last column, Table 8.2.2, Scheffe). The program generates and prints these expression in a form very close to the normal printed form. The method is from Scheffé, pp. 284-8.

5. Denominator for each source (FINDEN)

By the standard procedures, using the expected mean squares, the program determines the correct denominator (if any) for each source.

6. Sorting of sources for summary table (SORT)

The sources are sorted in a convenient order, combining all sources with the same denominator. This order is then used in printing the summary table.

7. Input of data (INPUTX)

The input data is read from the input device (the first parameter on the main program card). The grand means for each dependent variable are computed, ignoring missing (blank) data.

8. Storage of data for one dependent variable (READX)

This routine as well as all the remaining ones are executed in cycle once for each dependent variable. READX stored the data in core and checks that no data is missing in the design. The data is actually stored as deviations from the grand mean. This is done to improve accuracy. See Section 4.2.

9. Check of replication numbers (CELLN)

In the case of Class A and B designs, BALANOVA 5 checks whether the call frequencies are equal, proportional or non-proportional.

10. Computation of sum of squares (SSEQU,SSPROP,XMEAN)

The marginal means and sums of squares for each legal source are calculated.

11. Computation and printing of final summary table (FISHER,FPRINT)

These calculations are made in the standard way.

4.2 Some comments on accuracy of computation

An attempt was made in the design of this program to eliminate the largest sources of computational inaccuracies that can occur in analysis of variance calculations.

Consider a one-way analysis with the following data:

	Group 1	Group 2	Group 3
	8.77	8.88	8.96
	8.79	8.90	8.99
	8.80	8.90	9.00
	8.82	8.91	9.02
	8.82	8.91	9.03
Means	8.80	8.90	9.00

Sums of squares computations are generally made as the sums and differences of two or more terms. In this example, the exact calculations would be

$$\text{SS between} = 1188.2500 - 1188.1500 = 0.1000$$

$$\text{SS within} = 1188.2554 - 1188.2500 = 0.0054$$

Note that these answers each have a string of zeros following the given digits since they are exact. However on a computer, with about 8 digit accuracy, the differences would only be accurate to about four decimals due to the cancellation of all the higher order digits by subtraction.

This is illustrated by a calculation using the previous analysis of variance program in SOUPAC. SOUPAC's answers were:

SS between = 0.10000610 (5 significant digits)

SS within = 0.0053863525 (2 significant digits)

Note the large errors in these SS. Even worse errors can occur in other data.

The whole problem could be avoided by accumulating a true sum of square that is, by adding positive numbers to form each SS rather than taking a difference of two large numbers. However this procedure was rejected because it is extremely slow.

The following procedure is used in BALANOVA 5 and it is very effective. The data are internally transformed to deviations from the grand mean. This is why the grand means are computed in subroutine INPUTX before the deviations are actually stored in the memory in subroutine READX. When the deviations are used the individual terms which are added and subtracted to give each SS are now numbers of approximately the same size as the SS itself. This means that the number of significant digits in the SS is large even if the grand mean is large. In the example given above, the deviation scores are:

	Group 1	Group 2	Group 3
	-.13	-.02	+.06
	-.11	.00	+.09
	-.10	.00	+.10
	-.08	+.01	+.12
	-.08	+.01	+.12
Means	-.10	.00	+.10

and the SS are computed as

SS between = 0.10000000 - 0.00000000
= 0.10000000

SS within = 0.10540000 - 0.10000000
= 0.00540000

The actual results produced by BALANOVA 5 were

SS between = 0.099999998 (8 significant digits)

SS within = 0.0053999992 (7 significant digits)

Note the great improvement in accuracy.

As a final feature of BALANOVA 5, the approximate number of significant digits in each SS is calculated and printed alongside each SS. These numbers should not be interpreted exactly but only as a warning when they are small. The approximate number of significant digits is calculated in the following way

(a) Find the largest term, in absolute value, entering into the calculation of the SS. In the example above, the largest term in SS within is the first term (0.10540000).

(b) Take the ratio of this largest term to the SS itself. In the example, this ratio = 0.10540000/0.00540000.

(c) The approximate number of significant digits is then = $8.0-\log_{10}$ (ratio). In the example, this is 8.0 - 1.38 = 6.62 which is printed by BALANOVA 5 as 7, a pretty good estimate.

Note that the number of significant digits printed by BALANOVA 5 reflects the loss of accuracy in the computation of the SS from two or more terms. It does not reflect loss of accuracy due to computation of the terms themselves.

4.3 Error conditions

BALANOVA 5 makes a detailed check to insure that the design is legal, that none of the computer storage arrays are exceeded and that all the data corresponds to cells within the specified design. The following general types of errors are distinguished and corresponding error messages are printed giving detailed instructions about how to correct the error.

1. One of the restrictions on program size has been exceeded. These restrictions are:

(a) maximum number of factors = 10

(b) maximum number of legal sources = 100

(c) maximum size of X-storage array (used for data, means and cell numbers) depends on region

(d) maximum number of dependent variables = 200

(e) maximum number of sigma-squared terms in any one expected mean square = 10

2. The factor specification cards are incorrect or inconsistent. This is, the design is illegal. The checks made are:

(a) all nested factors must be listed as a factor.

(b) no factor may be nested within itself.

(c) at most one factor can be the replication factor. Furthermore, the replication factor must be nested in at least one other factor and no factor can be nested in the replication factor.

(d) the factor type must be fixed or random.

(e) the maximum number of levels for each factor must be more than one.

(f) there must be at least one denominator term in the analysis of variance summary table. If this is not the case it is probably due to no factor being designated as a random factor.

3. A Data Card has a level set which exceeds the limits stated in the maximum number of levels on the Factor Specification Cards.

4. Once the data for a dependent variable has been read in, a detailed check is made to insure that all cells in the design are filled. If one cell is not, the calculation for that design is deleted and the program moves on to the next dependent variable after printing sufficient information for the user to locate the missing datum. An additional check for Class B and C designs is made to insure that data for a given subscript set is not read in twice. If two data cards specify the same level set, a comment is made to this effect and the calculations for the dependent variable are deleted. Note that both the checks mentioned in this paragraph are made independently for each dependent variable and are made after the missing data (blank fields) for that dependent variable have been deleted. Errors referred to in this paragraph are not fatal and the program proceeds to the next dependent variable.

5. About one dozen other checks are made. They should always be passed satisfactorily since the design is first checked as above. These additional checks were inserted to assist in debugging the program and if one of them fails it indicates a remaining error in the program. A printed message is made to this effect in these cases.

4.4 Program Checkout

BALANOVA 5 has been checked on a large number of designs. Among these, the following calculations were reproduced by BALANOVA 5:

1. Lindquist, p. 266, Class A.

2. Winer. Table 7.8-3 (p. 376), both least-squares and unweighted means, Class B.

4.5 Number of levels of the replication factor

The rules in Section 1.5 and Chapter 3 are strict in the sense that, if they are followed, BALANOVA 5 will execute correctly. However the rules may be relaxed or ignored in the case of the number of levels of the replication factor and it is sometimes convenient to do so.

In Class A designs, any number of levels of the replication factor may be punched on the Main Parameter Card provided the number is ≥ 2. This

is so because in Class A designs only cell means are stored and the program does not check the replication number anyway. This rule relaxation is useful when it is inconvenient for the user to calculate ahead of time how many subjects are in each cell.

In Class B designs, any number of levels of the replication factor may be punched on the Factor Specification Card provided

(a) the number is $\geq$ the maximum number of replications in any one nest, and

(b) the number is not so large that the restriction on the size of the X matrix is exceeded.

Again this rule relaxation saves the user from having to know the maximum number of replications before using BALANOVA 5, provided he knows an upper limit. The restriction on the size of the X matrix will not often be exceeded. However, the user has not been informed, in this manual, how to estimate the size of the X matrix needed, since this limit is complicated to specify.

4.6 Restrictions

1. No latin-squares designs may be run using Balanova 5.

2. No partially crossed designs may be run.

3. One and only one replication factor is allowed but one may use random factors as needed to a maximum of 10 total factors.

4. Missing data may not cause a cell to disappear.

5. Factor levels on all but the replication factor must be numbered 1 through the maximum, without skips.

4.7 References

Hays, W. L.: Statistics for Psychologists. New York: Holt, Rinehard and Winston, 1963.

Lindquist, E. F.: Design and Analysis of Experiments in Psychology and Education. Boston: Houghton Mifflin, 1953.

Scheffé, H. A.: The Analysis of Variance. New York: Wiley, 1959.

Winer, B. J.: Statistical Principles in Experimental Design. New York: McGraw-Hill, 1962.

APPENDIX A-KEY TO DESIGNS IN WINER AND LINDQUIST

To facilitate the use of Balanova 5 in conjunction with Winer and Lindquist, a large number of designs from these two books are listed below. All designs previously described in Chapter 2 are also cross-referenced below for convenience.

Winer, Chapter 3 — Class A

See Section 2.1.1.

Winer, Chapter 4, pp. 111-116 — Class C

See Section 2.3

Winer, Chapter 5, pp. 184-187 — Class B

See Section 2.2.2.

Winer, Chapter 5, pp. 188-189 — Class B

The design on p. 188 is a hierarchical design and has program cards (for n = 15):

```
BALANOVA (C)(4)(1)(2)(2)(2)(15).
(0).
(0)(0)(1).
(0)(0)(1)(2).
(1)(1)(1)(2)(3).
END P
```

The design on the top of p. 189 is identical to the design on p. 186 (Table 5.12-1) except for relabelling of factors. See Section 2.2.2.

The design on the bottom of p. 189 is actually a repeated measures design.

Winer, Chapter 6, pp. 233-238 — Class A

See Section 2.1.2.

Winer, Chapter 6, pp. 241-244 — Class A

See Section 2.1.2.

Winer, Chapter 6, pp. 252-257 — Class A

This is a 2 x 3 x 2 factorial design. Factor A is educational level (fixed). Factor B is training method (fixed). Factor C is instructor. Factor C may be considered to be either fixed or random. See the discussion in Winer, p. 253. Depending on the final choice of type for Factor C, the F ratios calculated by Balanova 5 will differ. Balanova 5 indicates what denominator was used for each F ratio.

```
BALANOVA (C)(4)(1)(2)(3)(2)(10).
(0).  FACTOR A
(0).  FACTOR B
(1).  FACTOR C
(1)(1)(1)(2)(3).  FACTOR D
END P
```

Note the SOUPAC feature that program cards can be commented, thus the names for factors used by Balanova 5 are given as comments and are in agreement with the notation used in the description above.

Winer, Chapter 6, pp. 283-287 — Class A

Balanova 5 accepts designs with all factors having two levels. Of course, Balanova 5 uses its regular computational method rather than any special technique for these designs.

Winer, Chapter 6, pp. 287-288 — Class A

The Wulff and Stolurow design and the Gordon design would both have program cards similar to those in Section 2.1. Note that Gordon considered that both main factors were random. Balanova 5 would make the correct F tests, i.e. the main effects are tested against the interaction mean square and the interaction is tested against the within cell (subject) mean square. If Winer's recommendation is followed, however, both of the factors should be considered fixed.

Winer, Chapter 6, p. 289 — Class C

See Section 2.3.

Winer, Chapter 6, pp. 289-291 — Class C

The last three designs in this section of Winer are repeated measures designs but the subject factor is not nested in any factor and hence the design must be a Class C design. The Bamford and Ritchie design has program cards:

```
BALANOVA (C)(3)(1)(3)(4)(9).
(0).
(0).
(1).
END P
```

The pooling of the subject interactions would have to be done by hand.

The Geranthewohl et al and Jerison studies are very similar except that the subject interactions were not pooled.

Winer, Chapter 6, pp. 291-297 — Class A

The special least squares computation for unequal cell frequencies described in Winer cannot be performed by Balanova 5. The data may be input to Balanova 5, however, and an unweighted means analysis will be performed. The program cards are:

```
BALANOVA (C)(3)(1)(2)(4)(27).
(0).
(0).
(1)(1)(1)(2).
END P
```

Note carefully, though, that Winer states that in this particular case the cell frequencies are, in a real sense, an integral part of the design and a least squares analysis is more appropriate than an unweighted means analysis.

Winer, Chapter 7 — Class B

See Section 2.2.1.

Lindquist, Chapter 5, p. 145 — Class C

The teatments x levels design with one observation per cell is a Class C design. Suppose there are 4 treatments and 10 levels. Then the program cards are:

```
BALANOVA (C)(2)(1)(4)(10).
(0).
(1).
END P
```

The levels factor must be random for there to be a test of the treatment effect.

Lindquist, Chapter 5, pp. 151-152 — Class A

Treatments x levels design (with more than one observation per cell) are Class A designs with three factors, namely, treatments, levels and subjects. Parenthetically, note that Lindquist's "levels" refers to a factor name and not to particular values of this factor, called "levels" throughout this writeup.

The exercise on pp. 151-152 has the program cards:

```
BALANOVA (C)(3)(1)(2)(3)(5).
(0).
(0).
(1)(1)(1)(2).
END P
```

The two main effects are test score, factor 1 and level of ability, factor 2. Subjects (factor 3) are nested within these two factors and subjects is the replication factor.

Note Lindquist's comment (p. 141) that the levels factor is not a random factor and hence the within cell mean square (MS_{Subjects}) is the correct error term for the test, ability and test x ability sources. Balanova 5's calculations agree with Lindquist.

Lindquist, Chapter 6 — Class C

The treatments x subjects design is analyzed in exactly the same way as the treatments x levels design with one observation per cell as described above (Lindquist, Chapter 5, p. 145).

Lindquist, Chapter 7 — Class B

The groups within treatments designs discussed in this chapter of Lindquist are simply hierarchical designs like the hospitals within drugs example in Section 2.2. The treatments are drugs and the groups are hospitals. See the program cards previously given in Section 2.2.

Lindquist suggests that in some cases, even of proportionality, it is desirable to use the unweighted means analysis. This can be done by the override provided by main parameter 14. Of course if the cell frequencies are not proportional, the unweighted means will be performed anyway by Balanova 5.

Lindquist also suggests (p. 182) that the groups may be considered as random samples from a population of groups, in which case the group factor should be listed as of random type although the tests will not actually change. Here is another case where the test is the same even though the interpretation of the results will differ depending on what assumptions are made.

Lindquist, Chapter 8 — Class A

```
BALANOVA (C)(3)(1)(4)(5)(10).
(0).  FACTOR A
(1).  FACTOR B
(1)(1)(1)(2).  FACTOR C
ENDP
```

These program cards would be fore the case of 4 treatments (Factor A), 5 replications (Factor B) and at most 10 subjects in each treatment - replication group. There are 5 x 4 = 20 groups in all. The tests that Lindquist suggests are based on factor B being a random factor. These tests are:

(a) test MS_A against MS_{AB} (p. 191)

(b) test MS_{AB} against MS_C (p. 197)

These tests would be automatically performed by Balanova 5.

To further comments should be made. Lindquist suggests, on the bottom of p. 196, that unweighted means must be used for the test (a) above even in the case of proportional designs. If this advice is followed the special override in Balanova 5 must be used. See Main parameter 14. Secondly, the pooled test suggested in the middle of p. 196 would have to be done by hand after the analysis by Balanova 5 had been completed. See, however, Scheffé's chapter on mixed models for a detailed discussion of these problems.

Lindquist, Chapter 9 — Class A

Chapter 9 contains a discussion of the general two-factor design of which the treatments x levels design is just a special case. The examples are similar to those in Winer and will not be repeated. Note that Lindquist p. 215, discusses the possibility that, in a design with factors A and B, if A is random, MS_{AB} is the correct denominator for testing factor B. This is, of course, done by Balanova 5 if factor A is listed as of random type in the program cards.

Lindquist, Chapter 10, pp. 226-228 — Class A

See Section 2.1.2.

Lindquist, Chapter 10, pp. 230-237 — Class A

This is a random replications of a two-factor design and the following program cards:

```
BALANOVA (C)(4)(1)(a)(b)(c)(d).
(0).  FACTOR A
(0).  FACTOR B
(1).  FACTOR C
(1)(1)(1)(2)(3).  FACTOR D
END P
```

Note that both factors C and D are random. The number of levels for A, B and C are a, b and c which would be integers in an actual example. The number of levels for D (d in the above example) would be the maximum number of replications in any one cell. The tests given in Lindquist will be carried out by Balanova 5. That is, the interaction AB will be tested against MS_{ABC}, ABC, AC and BC will be tested against $MS_{within} = MS_D$, A is tested against MS_{AC} and B against MS_{BC}.

Lindquist, Chapter 10, pp. 237-238 — Class C

The treatments x treatments x subjects design is a Class C design. It is identical to the Bamford and Ritchie design described above (Winer, Chapter 6, pp. 289-291).

Lindquist, Chapter 10, pp. 238-243 — Class A

The designs on these pages are similar to designs discussed previously. Care must be taken to decide which factors, if any, are random. The computed mean squares are not affected by this choice, but the choice of denominators is.

Lindquist, Chapter 13 — Class B

Three of Lindquist's designs, Types I, III and VI, are acceptable to Balanova 5 as Class B designs.

Type I has program cards:

```
BALANOVA (C)(3)(1)(4)(3)(10).
(0).
(0).
(1)(1)(2).
END P
```

This table corresponds to the chart on the top of p. 268 where the maximum of n_1, n_2, and n_3 is 10.

Type III has program cards:

```
BALANOVA (C)(4)(1)(4)(2)(3)(40).
(0).  FACTOR A
(0).  FACTOR B
(0).  FACTOR C
(1)(1)(2)(3).  FACTOR D
END P
```

This table corresponds to the chart on p. 282 where the maximum number of replications in a BC cell is 40.

Type VI has program cards:

```
BALANOVA (C)(4)(1)(2)(3)(4)(10).
(0).     A
(0).     B
(0).     C
(1)(1)(3).  D  SUBJECTS
END P
```

This table corresponds to the chart on p. 292 where the maximum number of replications in any level of C is 10.

APPENDIX B-QUICK REVIEW OF PARAMETER CARDS

BALANOVA 5

Parameter Number	Description
1	Input address. CARDS, SEQUENTIAL 1-15.
2	Number of factors.
3	Number of dependent variables.
4-13	Number of levels of factors.
14	1 if desire unweighted means.
15	1 to suppress means.

Subparameters

1	1 if random factor.
2	1 if is replication factor.
3-11	Factors in which this factor is nested.

END PROGRAM is required.

CLASSIFICATION

I. General Description

The CLASSIFICATION program is designed to measure individuals against previously determined groups in order to determine probable group membership. The classification is done in a reduced test space derived from discriminant analysis. The method of classification is based on the premise that a group is totally described by its mean (or centroid) and dispersion; the individual's relation to each group is determined by a χ^2 which indicates how many members of the group are farther from the centroid than he and a Bayesian probability of membership in the group based on this χ^2. For each individual, the χ^2 and probability for each group are given; the user then applies a decision rule of his choice for assigning individuals to groups.

Since analysis is to be performed in a reduced space, the means and dispersion of each group must also be reduced to this space. The CLASSIFICATION program performs these reductions.

The calculation of probabilities requires the specification of the number of members in each group against which the individual is being compared. They may be the numbers actually in the groups used for finding experimental means and standard deviations or the number of individuals from the total group being tested who are to be assigned to each group. These numbers are specified along with the number of discriminant functions as an input vector.

In every case the input is expected in the form in which it is output by the DISCRIMINANT ANALYSIS program.

I. Formulas and Calculations

The following formulas are given in terms of matrix arithmetic except for divisions among singletons. Dimensions i, j, k are respectively for variables, functions and groups.

V (with dimensions i, j): discriminant vectors (input) for i variables and j functions

$\bar{X}$ (with dimensions k, i): group means for k groups (input)

C (with dimensions k, j) = $\bar{X} \cdot V$: centroids in reduced space. Let $\hat{C}$ (vector dimensioned j) be C for one group.

The following are calculated for each group, k:

D (with dimensions i,i): dispersion matrix for group k (input)

$\hat{D}$ (with dimensions j,j) = $V' \cdot D \cdot V$: reduced dispersion matrix for group k

$\hat{D}^{-1}$ (with dimensions j,j): inverse of reduced dispersion matrix for group k

G (singleton): group sample size (input)

R (singleton) = G/determinant $\hat{D}$: ratio for group k

The following are calculated for each subject of raw data entered:

S (vector dimensioned i): row of data for a subject (input)

For each group k and each subject

$\chi^2 = d'\hat{D}^{-1}d$ where $d = S \cdot V - \hat{C}$

$P_1 = R \cdot e^{-\chi^2/2}$, where e is the base of the natural logarithm

Then for the subject $P_2 = \sum_k P_1(k)$

Probability of membership in group k: $\hat{p}(k) = P_1(k)/P_2$

III. Parameters

The program name CLASSIFICATION is followed by these parameters on the program call card:

Parameter Number	Description
1	Input Address of discriminant vectors, V. The vectors are expected as columns. CARDS or SEQUENTIAL 1-15.
2	Input Address of means, $\bar{X}$. The means for a given group are expected as a row. CARDS or SEQUENTIAL 1-15.
3	Input Address of dispersion matrices, D. The dispersion matrices are expected in a vertically augmented form. CARDS or SEQUENTAL 1-15.
4	Input Address of individual scores, S. CARDS or SEQUENTIAL 1-15.
5 Ω	Output Address for inverse of dispersion matrix. PRINT or SEQUENTIAL 1-15.
6	Output Address for χ^2 and probabilities for each subject. SEQUENTIAL 1-15 and/or PRINT. (Output on SEQUENTIAL is in the form N_i, X_{il}, ...X_{in}, Y_{il}, ...Y_{in} where N_i = sequential subject number in groups, X_{ij}^i = j^{th} probabilities for i^{th} subject, Y_{ij}^{ij} = j^{th} χ^2 for i^{th} subject)
7	Number of groups.
8	Number of variables.

Parameter Number	Description
9	Input Address of number of discriminant functions, and group sample sizes, G, as a single row. CARDS or SEQUENTIAL 1-15.*

Ω Output may be punched directly. See SOUPAC Manual, section of input and output.

* Must be a row vector (1 observation) with the number of discriminant functions as the first variable. The form output by DISCRIMINANT ANALYSIS program may be used. If coming from CARDS, this data deck should be first, followed by any other card decks in the order listed in the parameters.

IV. References

Cooley, William W. and Lohnes, Paul R., Multivariate Procedures for the Behavioral Sciences, Chapter 7, John Wiley & Sons, Inc., New York, 1962.

DISCRIMINANT ANALYSIS

I. GENERAL DESCRIPTION

Suppose that we have k populations (groups) and p measures (variables) on each member of each population. We want to test the hypothesis that our groups are significantly different on the entire set of variables. This one-way multivariate analysis of variance hypothesis is tested by this program. The program then locates the dimensions (discriminant functions) along which the group differences are maximum. Thus, we need some function to transform the p variates into a smaller set of independent measures which will indicate the differences between the groups. The DISCRIMINANT ANALYSIS program finds the independent linear functions of the variables which maximally discriminate between k populations (groups input). The results from this program, namely the discriminant functions, may be used in the CLASSIFICATION program to determine the probability that any subject belongs in any group. Also, by looking at the coefficients of the functions, we can determine to what extent each of the p variates contributes to each function. In order to do this we need to determine the coefficients of the functions such that the ratio of variances between groups to the variances within groups is maximized, i.e. the differences between groups are to be large relative to the differences within groups.

II. CALCULATIONS AND FORMULAS

In matrix terms, we are trying to maximize the ratio

$$\lambda i = \frac{f_i' A f_i}{f_i' W f_i}$$

where f_i is the eigenvector associated with the i^{th} eigenvalue λi of $W^{-1} A$, A = the covariance matrix between means,

$$a_{ij} = \sum_{g=1}^{k} N_g (\bar{X}_{ig} - \bar{X}_i)(\bar{X}_{jg} - \bar{X}_j)$$

and W = the covariance matrix within classes,

$$w_{ij} = \sum_{g=1}^{k} \left[\sum_{n=1}^{N_g} (X_{ign} - \bar{X}_{ig})(X_{jgn} - \bar{X}_{jg}) \right]$$

where k = number of groups, N_g = number of subjects in group g, N = total number of subjects, and i and j run from 1 to p, where p = number of variables.

To find the maximum, we derive from the partial derivatives of that ratio, the matrix equation

$$(W^{-1} A - \lambda I) F = 0$$

where F is the matrix of eigenvectors. The eigenvectors are the coefficients of the discriminant functions. The relative sizes of the eigenvalues indicate the extent to which the associated discriminant functions distinguish

among the groups. The percentage of the total discriminating power of the variables contained in the j^{th} discriminant function is represented by

$$100\left(\frac{\lambda_j}{\sum_{i=1}^{N} \lambda_i}\right)$$

(N should be the smaller of k-1 and p)

In addition to obtaining the eigenvalues and discriminating coefficients, the program will compute scaled vectors to show the relative contributions of the variables to the discriminant function by

$$f_{ij}' = (w_{ii})^{1/2} f_{ij}$$

III. INPUT DATA

Input to the DISCRIMINANT ANALYSIS program consists of two or more data groups. Each data group consists of a set of observations on two or more variables. All of the groups must contain observations on the same set of variables. The groups may be input as separate card decks (each preceded by a DATA format card and followed by an END# card), as data groups located on separate temporary storage areas, or as a mixture of data groups on card decks and data decks on temporary storage areas. See section of examples.

The discriminant functions can be computed either from raw data or from W and T matrices, where each is a separate card deck or input file. See Output section for description of W and T.

IV. SIGNIFICANCE TESTS

The measure of significance calculated in the DISCRIMINANT ANALYSIS program is a Wilks' lambda (likelihood ratio test statistic). This is a test of the discriminating power of the test battery. It tests the hypothesis that the population centroids (mean vectors) are equal for the k groups. The Wilk's lambda is a function of the roots of $W^{-1}A$ and is of the following form:

$$\Lambda = \prod_{i=1}^{r} \frac{1}{1+\lambda_i}$$

where r is the lesser of k-1 and p. In matrix terms this criterion is defined in the following manner:

$$\Lambda = \frac{|W|}{|T|}$$

where $|W|$ and $|T|$ are determinants

W is the pooled within groups deviation score cross-products and T is the total sample deviation cross products matrix. As $|T|$ increases relative to $|W|$ the ratio decreases in value with an accompanying increase in the confidence that the group centroids are not equal.

An F ratio which yields an approximate test of the significance of the Wilks' lambda is calculated and printed.

$$F = \left(\frac{1 - y}{y}\right)\left(\frac{ms + 2\lambda}{2r}\right)$$

where $s = \sqrt{(p^2q^2 - 4)/(p^2 + q^2 - 5)}$ $\qquad q = k - 1$

$m = n - (p + q + 1)/2$ $\qquad n = N - 1$

$\lambda = -(pq - 2)/4$ $\qquad N$ = total number of subjects

$r = pq/2$ $\qquad k$ = number of groups

$y = \Lambda^{1/2}$ $\qquad p$ = number of variables

The degrees of freedom to be used with the F value printed in the output are printed and are labeled F1 (degrees of freedom for the numerator) and F2 (degrees of freedom for the denominator) and equal 2r and $mx + 2\lambda$, respectively.

V. OUTPUT

The output consists of the following:

1. Means of input variables for each group and group sample size (Parameter Number 8)

2. A dispersion matrix for each group. (Parameter Number 9)

3. The total sample deviation score cross-products matrix

$$t_{ij} = \sum_{n=1}^{N} (X_{in} - \bar{X}_i)(X_{jn} - \bar{X}_j)$$

where i and j range over the variables. This matrix is the sum of the A and W matrix described in section I. This is the T matrix referred to in Parameter Number 7. The diagonal of this matrix contains the sums of squares. (Parameter Number 6)

4. The pooled within-groups deviation scores cross-products matrix which is labeled W on the output. (Parameter Number 6)

5. The total number of subjects in all the groups combined. (Parameter Number 6)

6. The means and standard deviations of the variables across all groups. (Parameter Number 6)

7. The correlation matrix of variables over all groups. (Parameter Number 6)

8. The among groups cross-products of deviations of group mean from gra means weighted by group sizes. This matrix is labeled A matrix on the output. (Parameter Number 6)

9. The eigenvalues for the W^{-1} A matrix. (Printed)

10. The eigenvalues and percentage of variance explained by each additional eigenvalue. (Printed on output, automatically)

11. The trace of the W^{-1} A matrix. This is the sum of the eigenvalues. (Printed on output, automatically)

12. The discriminant functions (f_{ij}). The number of discriminant function will equal r where r is the lesser of the two values k-1 and p, where k = the number of groups and p = number of variables. (Parameter Number 1)

13. The group means on the discriminant functions. This is a k x r matrix formed by multiplying the group means on variables and the discriminant functions. The matrix may be used to determine the relative positions of the groups on the derived function. (Parameter Number 11)

14. The scaled vectors. These vectors are formed by multiplying the discriminant functions by the square roots of the diagonal of the W matrix described above. The scaled vectors show the relative contributions of the input variables to each of the discriminant functions. (Parameter Number 5)

15. The measures of significance described in Section IV. (Parameter Number 4)

VI. RESTRICTIONS

1. If raw data is input from cards, each group should be preceded by a DATA card and concluded with an END# card in accordance with SOUPAC conventions. If coming from sequentials data must be on separate sequential files.

2. The number of subjects in a group must be greater than or equal to the number of variables.

3. The W matrix may not be singular, that is deviations from group means may not be linearly dependent.

4. Variables may not be constant within a group.

VII. PARAMETERS

The DISCRIMINANT ANALYSIS program follows the program name on the main program card. Each parameter must be enclosed in parentheses. The parameters must appear in the order given below. If a parameter is not needed, do not punch anything between its parentheses. All parentheses after the last non-empty pair may be omitted.

Parameter Number		Description
1	Ω	Output address of discriminant functions (Matrix f_{ij}). SEQUENTIAL 1-15 and/or PRINT. (Needed for CLASSIFICATION.)
2		Number of variables
3		Number of groups
4		1 if desire significance measures printed.
5		1 if desire scaled discriminant vectors printed.
6		1 if desire intermediate results printed.
7		1 if input is W and T matrices instead of raw data.
8		Output address of group means on original variable and sample size (printed only).* SEQUENTIAL 1-15 and/or PRINT. (See Parameter 12) (Needed for CLASSIFICATION).
9	Ω	Output address of group dispersion matrices of original variables.* SEQUENTIAL 1-15 and/or PRINT. (Needed for CLASSIFICATION.)
10		N - total number of subjets in all groups combined. This parameter is left blank if raw data is input rather than W and T matrices.
11	Ω	Output address of group means on discriminant functions. SEQUENTIAL 1-15 and/or PRINT.
12		Output address of group sample sizes, needed for CLASSIFICATION.

* If W and T are input instead of raw data, group means and dispersion matrices are not printed. Means and dispersion matrices on discriminant functions are not computed in this case.

Ω It is possible to print in F format and/or punch the output from these parameters. If you need either of these options, see the section in the INTRODUCTION on INPUT and OUTPUT.

VIII. INPUT PROCEDURE

Input addresses of raw data groups or W and T matrices are listed on a $INPUT card following the main parameter card. W precedes T in the sequence when these are input. W and T may be output using a $OUTPUT card. Output order is the same as input.

If more raw data input groups are specified (parameter 3) than addresses on the $INPUT card, the last address specified is reused as many times as needed to provide that (parameter 3) number of groups. This is especially valuable if several decks of cards are input consecutively.

IX. SPECIAL COMMENTS

1. This program does not check for missing data. All blank spaces are read as zeros.

2. The user is cautioned against using the DISCRIMINANT ANALYSIS program without an understanding of the statistical technique used. See section of references.

3. Discriminant scores may be taken by matrix multiplication of raw data x discriminant functions (parameter 1).

X. EXAMPLES

A

```
/*ID
//  EXEC SOUP
//SYSIN  DD  *
MATRIX.
MOV(C)(S5).
END P
DIS(S1/P)(40)(4)(1)()()()(P).
$INP(C)(C)(S5)(C).
END SOUPAC
DATA(40)(40F2.0)
  .
  .  (1st Data Deck)
END#
DATA(40)(40F2.0)
  .
  .  (2nd Data Deck)
END#
DATA(40)(40F2.0)
  .
  .  (3rd Data Deck)
END#
DATA(40)(40F2.0)
  .  (4th Data Deck)
  .
  .
END#
/*
```

B

```
/*ID
//  EXEC  SOUP
//  SYSIN  DD  *
DIS(S1/P)(33)(4)(1)( )( )(1)( )( )(96).
$INPUT(C)(C).
END SOUP
DATA(20)(10X,5E14.7)
  .
  .  (Data for W)
  .
END#
DATA(20)(10X,5E14.7)
  .
  .  (Data for T)
  .
END#
/*
```

In Example A, four groups of data are being input with the first two groups coming from cards, the third from temporary storage on S5 and the fourth from cards. Discriminant functions are stored on S1 and printed and group means are printed. Significance measures are calculated for 40 variables input. In example B, W and T matrices are input and much the same results are obtained as for example A. Group means cannot be calculated.

```
/*ID  (accounting information )
//  EXEC  SOUPAC
//SYSIN  DD  *
DIS(S1)(15)(2)()()()()(S2)(S3)()()(S4).
$INP(C)(C).
CLA(S1)(S2)(S3)(C)(P)(P)(2)(15)(S4).
END S
DATA(15)(....format....)
  .
  .  (1st data deck)
  .
END#
DATA(15)(....format....)
  .
  .  (2nd data deck)
  .
END#
DATA(15)(....format....)
  .
  .  (3rd data deck)
  .
END#
/*
```

This illustrates the use of the DISCRIMINANT and CLASSIFICATION programs. The DISCRIMINANT program will save discriminant functions on S1; it will operate on 15 variables for each of two groups. It will output group means on S2 and group dispersion matrices on S3 and store group sample sizes on S4.

CLASSIFICATION, in turn, will read discriminant functions, group means, and group dispersion matrices from S1, S2, and S3 respectively. It will read the group to be classified from cards and print inverse of dispersion matrices and x^2 and probability. It will expect two sets of group means and dispersion matrices from 15 variables but 1 discriminant function (See Section V). Original group sample sizes are read from S4.

XI. REFERENCES

Bryan, J. F., The Generalized Discriminant Function: Mathematic Foundation and Computational Routine. Harvard Educational Review, (1951) 21:90-95.

Cooley, W. W. and Lohnes, P. R., Multivariate Procedures for the Behavioral Sciences. New York, Wiley, 1962.

Kendall, M. G., A Course in Multivariate Analysis. New York, Hafner, 1961, pp. 106-108.

References on Output:

Wilks' Lambda - first formula, page 119, Cooley and Lohnes.
F ratio - formula 4.1, page 62, Cooley and Lohnes.

T-TEST

I. General Description

The T-TEST program calculates a T coefficient or F ratio as described below:

Suboperation (1): Paired T-Test (also called correlated T-Test). Variables are in a row, variable one is paired with variable two, three with four, etc., and a paired T coefficient is calculated for each pair as follows:

$$t = \bar{d}/s_{\bar{d}}$$

$$\bar{d} = \sum_{i=1}^{N} [X_\sigma - X_\beta]/N_j \text{ , where } \sigma \text{ and } \beta \text{ are the } j^{th} \text{ pair of variables}$$

$$S_{\bar{d}} = N_j \sum_{i=1}^{N_j} [X_\sigma - X_\beta]^2 - \left(\sum_{i=1}^{N} [X_\sigma - X_\beta] \right)^2 / N_j{}^2 . f]^{1/2}$$

where f = degrees of freedom = N_j or N_j -1 as desired and N_j is the sample size for the j^{th} pair of variables.

Suboperation (2): Paired T-Test for all possible combinations of variables computed as in Suboperation (1).

Suboperation (3): Test of differences from a known population mean. A population mean must be provided for each column of data or the mean will be set to zero. Population means should be provided as a row vector. The following are calculated and printed for each variable:

t value: $t = (\bar{X} - \mu)/S_{\bar{x}}$

where μ = parameterized value or zero

Mean: $\bar{X} = \sum_{i=1}^{N} X_i/N$

Standard Deviation: $S.D. = \left(\frac{N \Sigma X_i^2 - (\Sigma X_i)^2}{N \times (d.f.)_i} \right)$

NOTE: N-1 is the usual degrees of freedom, but N may be specified.

Standard Error of Mean: $S_{\bar{x}} = \frac{S.D.}{\sqrt{N}}$

Suboperation (4): Test of differences from a known population mean for previously analyzed data: the mean ($\bar{X}$), standard deviation (S. D.), and sample size (N) as well as the population mean [see suboperation (3)] are

read in and the program computes the standard error of the mean and T for each trio of the X, S.D., and N read in that order. If R trios of data are used (R observations on rows occur) then R population means should be given. Calculations are the same as in Suboperation (3).

NOTE: A column of data of Suboperation (3) is reduced to a three item row here.

Suboperation (5): Test of differences between two or more group means taken pairwise: each group is located on a separate storage location or set of cards. The following are calculated for one variable, two groups:

t value: $t = (\bar{X}_i - \bar{X}_j)/S_{\bar{x}_i - \bar{x}_j}$ i and j are two groups

Mean: $\bar{X}_i = \sum_{k=1}^{N_i} X_{ik}/N_i$ for the variable over group i, N_i is sample size of group i

Standard Deviation: $(SD)_i = \left[\frac{N_i \sum_{k=1}^{N_i} X_{ik}^2 - \left(\sum_{k=1}^{N_i} X_{ik}\right)^2}{N_i \times (d.f.)_i}\right]^{1/2}$

where (d.f.) is N_i or $N_i - 1$

Pooled Estimate of Variance: $S^2 = [(SD)_i + (SD_j)]/(d.f.)$

where (d.f.) is $N_i + N_j$ or $N_i + N_j - 2$ for N or N-1 respectively

Estimate of Standard Error: $S_{\bar{x}_i - \bar{x}_j} = \left[\frac{S^2(N_i + N_j)}{N_i N_j}\right]$

Non-pooled Estimate of Standard Error:

$$S_n = [(S D)_i/(N_i^2 - 1) + (S D)_j/(N_j^2 - 1)]^{1/2}$$

Suboperation (6): One-way analysis of variance. The data to be compared are located on different storage units or sets of cards. Calculations are made as follows for each variable:

S = number of storage units = number of subgroups

N_i = number of observations in i^{th} subgroup, i=1,....., S, and for each i, j=1,....., N_i

X_{ij} = element in the j^{th} row of the i^{th} subgroup

$N = \sum_{i=1}^{S} N_i$ = total observations

$$\text{Total SS} = \sum_{i=1}^{S} \sum_{j=1}^{N_i} X_{ij}^2 - \left(\sum_{i=1}^{S} \sum_{j=1}^{N_i} X_{ij}\right)^2/N$$

$$\text{Between SS} = \sum_{i=1}^{S}\left[\left(\sum_{j=1}^{N_i} X_{ij}\right)^2/N_i\right] - \left(\sum_{i=1}^{S}\sum_{j=1}^{N_i} X_{ij}\right)^2/N$$

Within SS = Total SS - Between SS

Within D.F. = Total D.F. - Between D.F.

$$F = \frac{\frac{BSS}{BDF}}{\frac{WSS}{WDF}}$$

Suboperation (7): One way analysis of covariance. The experimental (dependent) variable comes first followed by 1 or more covariates. The dependent variable is adjusted to the set of covariates, not iteratively to one covariate at a time. Subgroups or factor levels are handled as in analysis of variance, Suboperation (6), i.e. as separate input decks or temporary storage locations. Statistics obtained are means and standard deviations, test of homogeneity of regression, F-ratio for covariance, and adjustment coefficients. For further discussion see Winer, p. 578.ff.

$$c_{ijk} = \sum_{N_k} X_{ik}X_{jk} - \frac{\sum_{N_k} X_{ik} \sum_{N_k} X_{jk}}{N_k}$$

deviation crossproducts for all variables X in group k (including experimental), X_{ik} and X_{jk} are two variables, N_k is number of subjects, $\sum_{N_k}$ is summation over group

$$C_k = \begin{array}{c|c|ccc} & y & x_1 & x_2 \cdots & x_m \\ \hline Z_k & & A_k & & \\ \hline A_k & & D_k & & \end{array}$$

where c_{ijk} is an element of C_k, y is experimental variable, $x_1, \ldots x_m$ are covariates

$$S_1 = \sum_k (Z_k - A_k' D_k^{-1} A_k)$$

= unpooled within group sum of squares

Group mean: $\bar{X}_{ik} = \frac{\sum_{N_k} X_i}{N_k}$ for group k

Group standard deviation: $s_{ik} = \sqrt{\frac{C_{ii}}{(d.f.)_k}}$

$$t_{ij} = \sum_{N} X_i X_j - \frac{\sum_{N} X_i \sum_{N} X_j}{N}$$

deviation cross products for all X overall

$$T = \begin{array}{c|c|ccc} & y & x_1 & x_2 \cdots & x_m \\ \hline Z & & A' & & \\ \hline A & & D & & \end{array}$$

where t_{ij} is an element of T, y is experimental variable, $x_1, \ldots, x_m$ are covariates

$S_5 = Z - A'D^{-1}A$

= overall sum of squares

Overall standard deviation $s_i = \sqrt{\frac{t_{ii}}{(d.f.)}}$

Correlation $r_{ij} = \frac{t_{ij}}{s_i s_j}$ Overall mean: $\bar{X}_i = \frac{\sum N X_i}{N}$

$$w_{ij} = \sum_k \left[N_k X_{ik} X_{jk} - \frac{\sum N_k X_{ik} \sum N_k X_{jk}}{N_k} \right] = \sum_k C_{ijk}$$

$$W = \begin{array}{c|c|ccc} & y & x_1 & x_2 \cdots & x_m \\ \hline & Z_w & & A'_w & \\ \hline & A_w & & D_w & \end{array}$$

where w_{ij} is an element of W,
y is experimental variable
$x_1, \ldots x_m$ are covariates

$S_2 = Z_w - A_w' D_w^{-1} A_w$

= adjustment sum of squares (pooled)

Then

S3 = S2 - S1
S4 = S5 - S2

$L_1 = N - (m+1)k$ where m = no. of covariates
k = no. of groups
N = overall sample
N_k = sample size for group k

$= \sum_k [N_k - (m+1)]$ c.f. Winer, p. 591

$L_2 = N - k - m$

$L_3 = m(k-1)$

$L_4 = k - 1$

Homogeneity of Regression

Source	Sum of Squares	Degrees of Freedom	Mean Square	F Ratio
pooled within	S3	L3	S3/L3	(S3/L3)/(S1/L1
unpooled within	S1	L1	S1/L1	

Analysis of Covariance

Source	Adj. Sum of Squares	Degrees of Freedom	Adj Mean Sq.	F Ratio
SS treatments	S4	L4	S4/L4	(S4/L4)/(
residual	S2	L2	S2/L2	

$B = D_w^{-1} A_w$

= adjustment coefficient for covariate

$y_{adj\ k} = \bar{y}_k + \sum_m [(\bar{x}_i - \bar{x}_{ik}) B_i] \quad i = 1,\ldots,m$

= adjusted group mean for experimental variable

I. References

Bryant, E.C., Statistical Analysis. New York: McGraw-Hill, 1960

Snedecor, G.W., Statistical Methods. Ames: Iowa State College Press, 1957.

Winer, B.J., Statistical Principles in Experimental Design. New York: McGraw-Hill, 1962.

I. Restrictions

Suboperations (5), (6), and (7), tests of differences and analysis of variance and covariance, require the data to be divided into 2 or more subgroups.

V. Parameters

T-TEST and ANALYSIS OF VARIANCE and COVARIANCE

Parameter Number	Description
1	Suboperations 1 - 7. (See above.)
2	Number of subgroups (if applicable).
3	0 - Count blanks as zeros 1 - Count blanks as missing data
4	0 - use N-1 as degrees of freedom 1 - use N as degrees of freedom (See Special Comments).
5	0 - pooled standard error in opt. 5 1 - non-pooled standard error (variances assumed unequal).

V. Special Comments

All T-TEST input is provided through a $INPUT card. Input addresses of subgroups for options 5, 6, and 7 are listed on a $INPUT card. (See section on SOUPAC Input/Output). See examples for illustration. If options 1 or 2 are used, provide only 1 input address on a $INPUT card. For option 3, provide two addresses, the first for the sample being tested, the second for the criterion means. For option 4 provide two input addresses, the first for the

means, standard deviations and sample size of the sample, the second for the criterion means. No second address with options 3 and 4 means a zero criterion mean is to be used.

If row vector of means is used this must be of length equal to number of variables.

Most work requires N-1 degrees of freedom. Option 7 does not check for missing data.

VI. Examples

1. A Complete Program

```
/*ID   [accounting information]
//  EXEC  SOUP
//SYSIN  DD *
T-T (5)(3)(1).
$INP(C)(C)(C).
END S
DATA (10)(10F2.0)
       [data cards for group 1]
END#
DATA (10)(10F2.0)
       [data cards for group 2]
END#
DATA (10)(10X,10F2.0)
       [data cards for group 3]
END#
/*
```

This is a complete SOUPAC program to do T-tests on ten variables and three groups taken pairwise, i.e. 1 vs 2, 1 vs 3 and 2 vs 3, thus 30 T's will result, printed in 10 tables. Note that data decks are stacked one behind another, each with its own DATA statement and END# card. A format appropriate to the data is given. The form of this is optional, but all groups should have the same number of variables. A check for missing data, coded blank or -0.0, will be made.

If the T-T card were replaced by T-T(6)(3)(1)., an analysis of variance would be performed yielding an F-ratio for each variable. Other cards could remain the same.

2. Other examples T-T and corresponding $INPUT cards follow. Note that input from S (sequential) units requires that data be stored there earlier in the <u>same</u> run, or permanently stored in the system. Data decks will always be called for in the order listed (unless otherwise specified).

```
T-T (1).                          Paired T-Test on data stored on S1.
$INP(S1).
```

```
T-T (2)( )(1)(1).
$INP(S15).
```
Paired T-Test on data stored on S15 doing test on all possible pairs checking for missing data and using N degrees of freedom.

```
T-T (3).
$INP(S1)(S3).
```
T-Test of population means on S1 against criterion means on S3.

```
T-T (5)(2).
$INP(S1)(S2).
```
T-Test of group mean of two groups, one on S1 and the other on S2.

```
T-T (6)(4)(1).
$INP(S1)(C)(C)(S3).
```
One-way analysis of variance over four groups, one on S1, two from cards, and one from S3, checking for missing data.

CORRELATIONS AND REGRESSION PACKAGE

BISERIAL CORRELATION

I. General Description

This program calculates the following coefficients for each combination of one dichotomous and one continuous variable.

Case totals:

$$\% \text{ cases in } p = N_p/N$$

$$\% \text{ cases in } q = N_q/N$$

$$\text{Total cases} = N$$

Mean:

$$\bar{X}_p = \Sigma X_p/N_p$$

$$\bar{X}_q = \Sigma X_q/N_q$$

$$\bar{X} = \Sigma X/N$$

Standard deviation:

$$S_p^2 = \Sigma X_p^2/N_p - \bar{X}_p^2$$

$$S_q^2 = \Sigma X_q^2/N_q$$

$$S^2 = \Sigma X^2/N - \bar{X}^2$$

Biserial r:

$$r = \frac{(X_p - X_q)\ pq}{S\ (.3989)\ h}$$

where p = percentage of cases in 0 category
q = percentage of cases in 1 category
h = height of the normal curve computed from normal tables

The program checks for missing data, and computes the above measures only for those cases where both dichotomous and continuous variables are present.

II. Type of Input

Data is read row-wise from either tape or cards. All dichotomous variables must be first in each row. They should be coded with 0 and 1

III. Parameters

The program call card requires 4 parameters after the program name, BISERAL R:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15.
2	Output Address of correlations. SEQUENTIAL 1-15 and/or PRINT.
3	Number of dichotomous variables.
4	Number of continuous variables.
5	0 - treat blanks as missing data 1 - count blanks as zeros

CANONICAL ANALYSIS

I. General Description

The CANONICAL CORRELATION program provides a multivariate test of the hypothesis that two sets of normally distributed variables are independent. The larger set of variables (the predictor variables) is considered to have q members, and the smaller set (criterion variables) has p memebers. This program also linearly transforms each set of variables into a new set of independent variables, (or dimensions) such that the first new predictor (a linear combination of the original predictors) has maximum correlation with the first new criterion variable. The second new predictor is maximally correlated with the second new criterion, and so on (with the constraint that each new variable is uncorrelated with the previous new variables derived from the same set of original variables.

Let the criteria set consist of the p variables $x_1,\ldots,x_p$ and the predictor of q other variates $x_{p+1},\ldots x_{p+q}$. Assume $p \leq q$. We then look for weighting matrices $U_{\alpha\beta}$ and W_{ab} such that

$$\zeta_\alpha = \sum_\beta U_{\alpha\beta} x_\beta \qquad \begin{matrix}\alpha = 1, 2, \ldots p \\ \beta = 1, 2, \ldots p\end{matrix}$$

$$\eta_a = \sum_b W_{ab} x_b \qquad \begin{matrix}a = p+1, p+2, \ldots p+q \\ b = 1, 2, \ldots p\end{matrix}$$

The variables ζ and η have the following properties:

1. They are standardized variables.

2. Within each set, the ζ's are independent and the η's are independent, i.e. within the set ζ_α (as α runs from 1 through p) and within the set of η_a (as a runs from 1 through q) the correlations are zero.

3. The correlation between any ζ and any η is zero except for p correlations $\lambda_1,\ldots,\lambda_p$.

The purpose of this program is to find the p correlations $\lambda_1,\ldots,\lambda_p$ and the weighting matrices $U_{\alpha\beta}$ and W_{ab}.

II. Formulas and Calculations

The correlation matrix R is first partitioned into:

A	C'
C	B

where A = correlation among predictors
B = correlation among criteria
C = correlation between predictors and criteria

Standardized regression coefficients: $\beta = A^{-1}C'$

Multiple R-squared: $R^2 = CA^{-1}C' = C\beta$

The following equation is solved for λ^2 and U

1) $(CA^{-1}C' - \lambda^2 B)U = 0$ where A, B, and C are as above, λ^2 = canonical r^2, and U = criteria weighting matrix

The solution is obtained by using the following derived forms:

2) $(B - \gamma I)\, h = 0$ where γ represents eigenvalues of B, I is identity, and h represents eigenvectors of B

3) $[(HD^{-1/2})'(CA^{-1}C')(HD^{-1/2}) - \lambda^2 I]v = 0$ where D is the diagonal matrix of λ, H is matrix of h vectors λ^2 represents eigenvalues of E and v represents eigenvectors of E, $E = (HD^{-1/2})'(CA^{-1}C')(HD^{-1/2})$

4) $U = HD^{-1/2}V$ where V is matrix of v vectors

$B = HDH'$ needed only to prove equivalency of 1) with 2)-4).

The Predictor weighting matrix: $W = \beta U F^{-1/2}$ where F is diagonal matrix of λ^2, i.e. elements $F^{-1/2}$ are $1/\lambda$ on the diagonal and off.

Wilks' Lambda: $\Lambda_j = \prod_{i=j}^{P} (1 - \lambda_i^2)$ for the j^{th} function

Chi-Square: $\chi_j^2 = \log_e (\Lambda_j)\left(\frac{p+q+1}{2} - n\right)$ where n is sample size

III. Input

Input to the CANONICAL ANALYSIS program consists of a correlation matrix. These variables include a set of predictor variables and a set of criterion variables. Either set may be first on the input data but there can be no mixing of the two types of variables on the input data. The TRANSFORMATION program may be used to reorder the variables if they are mixed on the card data deck.

IV. Significance Tests

Included in the printed output of the CANONICAL program is a Chi-square value for each of the eigenvalues λ^2 computed in the program. The chi-square values printed are determined from the Wilks' lambda values using the procedure outlined by Bartlett (See Section X). The chi-square values provide a test of the null hypothesis that the p variates are unrelated to the q variates. If there is at least one way in which a linear combination of the criterion variables

is correlated with a linear combination of the criterion variables this Chi-square value will be significant. The second Chi-square may then be examined. This Chi-square is a test of a second relationship after the first relationship has been removed. If this Chi-square is significant a second linear combination of the predictor variables is correlated with a second linear combination of the criterion variables. This process continues until the first non-significant Chi-square is found. All Chi-squares beyond that point will be non-significant.

V. Output

The output consists of the following:

1. The matrix of standardized regression coefficients. This is the matrix of coefficients which would be formed if the raw data used to calculate the correlation matrix input had been converted to standard scores. The predictor variables are on the rows of the matrix and the criterion variables are on the columns of the matrix.

2. A multiple correlation squared (R^2) for each of the criterion variables. The first R^2 value is the multiple correlation of the first criterion variable with the entire set of predictors variables. The second is for the second criterion variable with the set of predictors, etc.

3. A set of eigenvalues λ^2, correlations λ, Wilks' lambdas, Chi-squares, and degrees of freedom. (See Section IV) (Printed)

4. A matrix of criterion weights (Parameter Number 3)

5. A matrix of predictor weights (Parameter Number 4)

VI. Restrictions

A. Precalculated correlation matrices should be punched with sufficient accuracy; accumulated round-off can cause malfunctions and errors in results. Raw data and the CORRELATION program should be used whenever possible, or maximum accuracy preserved in punching.

B. The matrices A and B must both be non-singular.

C. Number of criteria (p) must be less than or equal to the number of predictors (q).

II. Parameters

The parameters for the CANONICAL ANALYSIS program follow the program mnemonic CAN on the main program card. Each parameter must be enclosed in parentheses. The parameters must appear in the order given below. If a parameter is not needed, do not punch anything between its parentheses. All parentheses after the last non-empty pair may be omitted.

Parameter Number		Description
1		Input Address (correlation matrix). CARDS or SEQUENTIAL 1 - 5.
2		Sample size of raw data needed for Chi-square. May be zero or blank if unknown.
3	Ω	Output Address of criterion weighting matrix. SEQUENTIAL 1-5 and/or PRINT.
4	Ω	Output Address of predictor weighting matrix. SEQUENTIAL 1-5 and/or PRINT.
5		Number of predictor variables.
6		Number of criterion variables. (Must be less than or equal to number of predictors).
7		Order of variable sets on input: 1 if predictors are first 2 if criteria are first
8		1 if want regression coefficients printed
9		1 if want multiple correlation squared (R^2) printed
10		Output Address of eigenvalues. Print is not valid.

Ω It is possible to print in F format and/or punch the output from these parameters. If you need either of these options, see the section in the Introduction on Input and Output.

VIII. Special Comments

This program does not check for missing data. All blank spaces are read as zeros.

IX. Examples

A

```
/*ID    [accounting information]
//  EXEC  SOUP
//SYSIN  DD  *
CANONICAL (CARDS)(50)
(PRINT)(PRINT)(15)(5)(1)(1)(1).
ENDS
DATA(20)(8F10.7)
  :
  :   punched correlations
  :
END#
/*
```

B

```
/*ID    [accounting information]
//  EXEC  SOUP
//SYSIN  DD  *
COR(C)(P)(S1/P).
CAN(S1)(50)(P)(P)(15)(5)(1)(1)(1).
ENDS
DATA(20)(20F4.0)
  :
  :     raw data
  :
END#
/*
```

Examples A and B illustrate the use of the program card for Canonical correlations. In these examples the input data consists of 15 predictor and 5 criterion variables and will be a card deck.

In example A the card deck is a punched correlation matrix; note that a possible format is 8 values to a card with 10 digits each. The format will be reused until 20 values are found per row. Thus the deck will be contained on 60 cards. The Canonical correlations program reads these cards.

In Example B the card deck is raw data, the format could be any string adequate to read the data. The data deck is read by the correlation program, which produces correlations and stores them on sequential 1 which Canonical reads.

The same calculations will be performed by each Canonical program (examples A and B). Fifteen variables are predictors and these are the first 15 in each row. The sample size of 50 was provided in order to get all significance tests. The printed output will be the canonical correlations, the criterion weighting matrix, the predictor weighting matrix, and significance tests.

X. References

For a discussion of the uses of canonical analysis, see Kendall, M.G., A Course in Multivariate Analysis, New York, Hafner, 1961, pp. 68-85 or Kendall, M.G., The Advanced Theory of Statistics, New York, Hafner, 1951, Vol. II, pp. 348-358.

For the derivation of the method used, see Anderson, T.W., An Introduction to Multivariate Statistics, New York, Wiley, 1958, pp. 288-296.

For significance test procedure, see Cooley, W.W. and Lohnes, P.R., Multivariate Procedures for the Behavioral Sciences, New York, Wiley, 1962, p. 37.

For details on the solution of eigenvalue matrices, see Johnston, J., Econometric Methods, New York, McGraw-Hill, 1963, pp. 95-103.

CORRELATION

I. General Description

The main purpose of the CORRELATION program is the calculation of Pearson product-moment correlations (hereafter referred to as correlations in this writeup). A correlation measures the linear dependency between two variables, and this program calculates a correlation for each pair of input variables. The square of a correlation, sometimes called the coefficient of determination, represents the proportional reduction in variance of one variable due to a linear relationship with another. Thus the coefficient of determination measures the strength of a linear relationship, or the proportion of variance accounted for by a linear rule.

The CORRELATION program automatically produces other types of correlation coefficients, because the calculations required are identical. Thus point biserial coefficients of correlation (often preferred to biserial correlation), phi coefficients (alternative to tetrachoric coefficients), and Spearman's rank order correlations can be readily obtained. Thus, if the input consists of dichotomous variables, the output will contain a mixture of phi's, point biserials, and ordinary correlations. (A point biserial correlation is a correlation between a dichotomous variable and a continuous variable). If the input to the correlation program consists of rank ordered data (ordinal), the output will be Spearman's rank order correlations. (See Walker and Lev, Chapter 11 for comparisons and comments on the above mentioned coefficients).

In the process of calculating the correlations, the means and standard deviations of the individual variables are computed, as are the cross-products and covariances between variables. After the correlations have been calculated, they are used to calculate the linear regression coefficients and corresponding intercept terms needed for predicting each variable from each other variable.

II. Input

Input to the CORRELATION program consists of a set of independent observations on two or more variables. The data is considered as a two-dimensional array (or matrix) of numbers with each column containing the observations on one variable, and each row consisting of one observation on each variable. If we use the letter X to represent the matrix of raw data, we let X_{ij} represent the i^{th} row (where $i = 1, 2, \ldots N$) and the j^{th} column (where $j = 1, 2, \ldots M$). In other words, we have N observations (rows) and M variables (columns) in our data matrix X.

III. Formulas and Calculations

The following formulas define certain statistics and illustrate their methods of calculating within the program. The subscript i refers to observations (or individuals) and runs from 1 to N. The subscripts j and k refer to variables, and they run from 1 to M.

Mean (of variable j) $= \bar{X}_j = \dfrac{\sum_{i=1}^{N} X_{ij}}{N}$

Covariance* (between variables j and k) $= C_{jk} = \dfrac{\sum_{i=1}^{N} (X_{ij} - \bar{X}_j)(X_{ik} - \bar{X}_k)}{N-1} = \dfrac{N \sum_{i=1}^{N} X_{ij}X_{ik} - (\sum_{i=1}^{N} X_{ij})(\sum_{i=1}^{N} X_{ik})}{N(N-1)}$

Standard Deviation* (of variable j) $= S_j = \sqrt{\dfrac{\sum_{i=1}^{N} (X_{ij} - \bar{X}_j)^2}{N-1}} = \sqrt{\dfrac{N \sum_{i=1}^{N} X_{ij}^2 - (\sum_{i=1}^{N} X_{ij})^2}{N(N-1)}} = \sqrt{C_{jj}}$

Correlation (between variables j and k) $= R_{jk} = \dfrac{C_{jk}}{S_j S_k} = \dfrac{N \sum_{i=1}^{N} X_{ij}X_{ik} - (\sum_{i=1}^{N} X_{ij})(\sum_{i=1}^{N} X_{ik})}{\sqrt{N \sum_{i=1}^{N} X_{ij}^2 - (\sum_{i=1}^{N} X_{ij})^2}\sqrt{N \sum_{i=1}^{N} X_{ik}^2 - (\sum_{i=1}^{N} X_{ik})^2}}$

From the equation $X_{ij} = B_{jk}X_{ik} + A_{jk}$, the program calculates

Linear Regression Coefficient (for predicting variable j from variable k) $= B_{jk} = R_{jk}(\dfrac{S_j}{S_k})$

Intercept (constant term in equation for predicting variable j from variable k) $= A_{jk} = \bar{X}_j - B_{jk}\bar{X}_k$

*NOTE: the sample covariances and sample standard deviations are unbiased estimates of the corresponding population parameters. The definitions given here follow the practice of many current statisticians. [See Anderson (1958) - Chapter 3 for example.]

IV. Significance Tests

If we assume that two variables (indexed by j and k) have a bivariate normal distribution, there is a test statistic for testing the hypothesis that the correlation in the population is zero (or equivalently that either regression coefficient is zero). Even for a relatively small sample size (N), this hypothesis can be tested using the t ratio:

$$t = \frac{R_{jk}\sqrt{N-2}}{\sqrt{1 - R_{jk}^2}}$$

with N - 2 degrees of freedom. Other types of hypotheses can be tested through use of the Fisher R to Z transformation. [See Hays (1963), pages 529 - 533 for example.]

V. Output

Output from the CORRELATION program may consist of any or all of the statistics from section III above, by using parameters 2 through 7. Any output from this program may be printed and/or output to temporary storage (SEQUENTIAL 1 - 5). The means, standard deviations, and the sample size (N) are output as a matrix with M rows (one for each variable) and three columns (the third column will have a constant value of N for all variables). Correlations, covariances, and cross-products are printed as lower triangular matrices, while the regression coefficients and intercepts are printed as square matrices. However, all five of these matrices are stored as square matrices.

VI. Restrictions

The CORRELATION program will accept an unlimited number of observations, but the number of variables is limited as noted in the section on PROGRAM LIMITS in the INTRODUCTION.

VII. Parameters

The parameters for the CORRELATION program follow the program name on the main program card. Each parameter must be enclosed in parentheses. The parameters must appear in the order given below. If a parameter is not needed, do not punch anything between its parentheses. All parentheses after the last non-empty pair may be omitted.

Parameter Number	Use or Meaning
1	Input Address of raw data (X matrix). CARDS or SEQUENTIAL 1-15.
2	Output Address for means, standard deviations, and sample size. SEQUENTIAL 1-15 and/or PRINT.
3	Ω Output Address for correlation matrix (R). SEQUENTIAL 1-15 and/or PRINT.
4	Ω Output Address for cross-products matrix. SEQUENTIAL 1-15 and/or PRINT.
5	Ω Output Address for covariance matrix (C). SEQUENTIAL 1-15 and/or PRINT.
6	Ω Output Address for matrix of regression coefficients (B). SEQUENTIAL 1-15 and/or PRINT.

7	Ω Output Address for intercepts (matrix A). SEQUENTIAL 1-5 and/or PRINT.
8	1 if last variable in each row is a weighting factor.

Ω - It is possible to print in F format and/or punch the output from these parameters. If you need either of these options, see the section in the INTRODUCTION on INPUT and OUTPUT.

VIII. Special Comments

1. This program does not check for missing data. All blank spaces are read as zeroes. If you have missing data, use the MISSING DATA CORRELATION program.

2. In the output matrices of regression coefficients and intercepts, the row number refers to the dependent variables, and the column numbers refer to the independent variables.

3. If a variable is constant, an error message will be printed and all correlations with that variable will be set to zero.

4. In order to have the program perform its calculations separately for subsamples of the data, see the section on CONTROL VARIABLES in the INTRODUCTION.

IX. Examples

1A

```
/*ID <accounting information>
//  EXEC  SOUP
//SOUP.SYSIN  DD  *
CORRELATIONS (CARDS)( )(PRINT).
END SOUPAC
DATA (6)(6F2.0)
  :
  :
END#
/*
```

1B

```
/*ID <accounting information>
//  EXEC  SOUP
//SYSIN  DD  *
COR (C)( )(P).
ENDS
DATA (6)(6F2.0)
  :
  :
END#
/*
```

Example 1A illustrates the usage of the CORRELATION program. Notice that all words are spelled out although this is unnecessary. Notice also that correlations are to be printed out, although the means and standard deviations are not. Example 1B will perform exactly the same computations as 1A, except that all instructions have been abbreviated to make keypunchi easier.

```
2
/*ID <accounting information>
//  EXEC  SOUP
//SYSIN  DD  *
COR (C\(P)(P/S1).
PRINCIPAL AXIS FROM (S1) TO (S2/P) WITH (10) FACTORS AND
(100) PERCENT OF THE VARIANCE TO BE REMOVED.
VARIMAX ROTATION FROM (S2) TO (PRINT).
ENDS
DATA (20)(10F4.0,5F6.2/10X,5F4.1)
  .
  .
  .
END#
/*
```

In the second example, the CORRELATION program first prints the means and standard deviations. Then it prints the CORRELATION matrix and stores it on SEQUENTIAL 1 (S1). The PRINCIPAL AXIS program then performs a principal components analysis and outputs 10 components to S2. VARIMAX then rotates these 10 components, using the VARIMAX criterion, and prints the results.

X. References

T. W. Anderson, An Introduction to Multivariate Statistical Analysis; John Wiley and Sons, Inc., 1958.

E. C. Bryant, Statistical Analysis; McGraw-Hill, 1960, pp. 113-135.

W. L. Hays, Statistics for Psychologists; Holt, Rinehart and Winston, 1960.

H. M. Walker and J. Lev, Statistical Inference; Henry Holt and Company New York, 1960.

MISSING DATA CORRELATION

I. General Description

The MISSING DATA CORRELATION program calculates the following coefficients for every combination of variables:

Mean: $X_i = \frac{\Sigma N_i}{N_i}$

Standard Deviation: $s_{xij} = \left[\frac{N_{ij}\Sigma(X^2_{ij}) - (\Sigma X_{ij})^2}{N_{ij}\ (N_{ij}-1)}\right]^{1/2}$

Covariance: $S_{ij} = \frac{N_{ij}\Sigma(XY_{ij}) - (\Sigma X_{ij})(\Sigma Y_{ij})}{N_{ij}\ (N_{ij}-1)}$

Correlation: $r_{ij} = \frac{S_{ij}}{Sx_{ij}Sy_{ij}}$

II. Restrictions

The maximum number of variables for this program is 100.

The input data to this program may come from any source conforming to SOUPAC. Output may be printed and the correlation matrix may be placed on any source conforming to SOUPAC.

III. Parameters

The parameters for the MISSING DATA CORRELATION program appear on the program card. They must follow the program name in the following order:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15. Default is CARDS.
2	0 - printing as usual 1 - printing is suppressed
3	Output Address of correlation matrix.
4	Output Address for sample sizes.
5	Coding for missing data; if left blank or if zero is entered, minus zero is used as check. It is NOT possible for this program to count true zeroes as missing data. This parameter must be enclosed in asterisks. Example: *99*.

NOTE: All output is in double precision.

IV. Special Comments

A. The user is warned against further processing of the correlations output by this program because the correlations do not necessarily come from the same sample.

B. For control breaks, data must be presorted on the control variables with the last variable changing fastest. The maximum number of contro variables is 30 Control variables begin on a new card with $C-B in c l and are enclosed in parentheses.

C. The correlation matrices can be stored in parameter 3 is a temporary storage address. However, if control breaks are also being used, only the first matrix corresponding to the first control break can be saved.

MULTIPLE CORRELATION

I. General Description

The MULTIPLE CORRELATION program calculates the following coefficients:
where n = sample size
m = sum of weights [$m = \sum_{i=1}^{n} w_i$]
p = number of independent variables (Parameter 2)
S = vector of standard deviations
$\bar{X}$ = independent variable means
$\bar{Y}$ = dependent variables means

Mean: $\bar{X}_j = \dfrac{\sum_{i=1}^{n} x_{ij}}{n}$

Raw Data Cross-Products: $X'X = \sum_{i=1}^{n} (x_{ij}x_{ik})$

Covariance: $c_{jk} = \dfrac{X'X}{(n-1)} - \dfrac{(\Sigma x_j)\ (\Sigma x_k)}{n(n-1)}$

Standard Deviation: $s_j = \sqrt{c_{jj}}$

Product Moment Correlation: $r_{jk} = \dfrac{c_{jk}}{s_j s_k}$

The correlation matrix is then partitioned as follows:

A	B
B'	C

where A is the independent variables correlation matrix. C is the dependent variables correlation matrix. And B and B' are the cross-correlation matrix.

Standardized Regression Coefficients: $\beta = A^{-1}B$

Deviation Covariance Matrix: $D = S(C-B'\beta)S' \dfrac{n-1}{[n-p-1]}$

This leaves the following matrix:

A^{-1}	β
B'	D

Deviation (Partial) Correlation Matrix: $D_{jk} = D_{jk} / (D_{jj} D_{kk})^{1/2}$

Regression Covariance Matrix: $RC = SB'\beta S' \left[\frac{n-1}{p}\right]$

Multiple Correlation: $R_j = (B'\beta)_j^{1/2}$

F Ratio: $F_{j\ n-p-1}^{\ p} = \frac{R_j^2}{1-R_j^2} \left[\frac{n-p-1}{p}\right]$

Standard Error of Estimate: $S_{e_j} = s_j[(1-R_j^2)\ \frac{n-1}{n-p-1}\ ]^{1/2}$

Unstandardized Regression Coefficient: $b_{j.k} = \frac{s_j}{s_k}\ \beta_{j.k}$

Dependent Variable Intercept: $b_{o.k} = \bar{Y}_k - \sum_{j=1}^{p} b_{j.k}\ \bar{X}_j$

Standard Error of Unstandardized Regression Coefficient: $s_{bj.k} = S_{e_k} [(X'X)_{jj}^{-1}]^{\frac{1}{2}}$

Standard Error of Standardized Regression Coefficients:

$$S_{\beta_{j.k}} = \frac{S_k}{S_j}\ S_{b_{j.k}}$$

T = Regression Coefficient/Standard Error: $T_{j.k} = \frac{\beta_{j.k}}{s_{\beta_{j.k}}}$

Predicted Dependent Variables: $y_k^* = b_{o.k} + \sum_{j=1}^{p} b_{j.k}\ x_j$

Deviations from Observed: $z_j = y_j - y_j^*$

Durbin-Watson Coefficient:[1] $$d_j = \frac{\sum_{i=2}^{n} (z_{j_{i-1}} - z_{j_i})^2}{\sum_{i=1}^{n} (z_{j_i})^2}$$

For reference to formulas and interpretations see:

E. C. Bryant, Statistical Analysis, New York, McGraw-Hill, 1960, pp. 198-224.

II. Restrictions

450 variables, 430 Dependent and Independent variables. (See Parameter 2)

The input data to this program may come from any source conforming to SOUPAC. Ouput will be printed and/or stored as indicated.

III. Parameters

The parameters for the MULTIPLE CORRELATION program appear on the program call card. (Most problems require only parameters 1, 2, 5, and 7, see example 1.) They must follow the program name in this order:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15, (See Special Comments for order of variables.)
2	Number of independent variables. Ind. var. + dep. var. + wt. var. + control var. $\leq$ 450. Ind. var. + dep. var. $\leq$ 430.
3	Output address of predicted dependent variables. SEQUENTIAL 1-15 and/or PRINT.
4	Output address of deviations from actual. SEQUENTIAL 1-15 and/or PRINT.
5	Output address of Means and Standard Deviations. SEQUENTIAL 1-15 and/or PRINT. 1st column contains Means. 2nd column contains Standard Deviations. 3rd column contains Sum of Weights.[2] 4th column contains Sample Size.

Parameter Number	Use or Meaning
6	Output address of coefficients (unstandardized). SEQUENTIAL 1-15, PRINT is default. Coefficients for M independent and N dependent variables are written as N rows with N + M + 1 columns each. The i^{th} row contains in order the i^{th} intercept term, the M coefficients for the i^{th} dependent variable, a -1 in the M + i + 1 location, and 0's for all other locations. This format is compatible with the ECONOMETRICS REDUCED FROM AND RESIDUAL ANALYSIS program.
7	Ω Output address of correlation matrix. SEQUENTIAL 1-15 and/or PRINT.
8	Ω Output address of raw data cross-products matrix. SEQUENTIAL 1-15 and/or PRINT.
9	Ω Output address for covariance matrix. SEQUENTIAL 1-15 and/or PRINT.
10	Ω Output address of deviation covariance matrix. SEQUENTIAL 1-15 and/or PRINT.
11	Ω Output address of deviation (partial) correlation matrix. SEQUENTIAL 1-15 and/or PRINT.
12	Ω Output address for regression covariance matrix. SEQUENTIAL 1-15 and/or PRINT.
13	Output address of Durbin-Watson and second, third and fourth powers of sums of deviations. SEQUENTIAL 1-15 and/or PRINT. Row 1 Durbin-Watson Coefficients. Row 2 $\Sigma(y-y^*)^2$ Row 3 $\Sigma(y-y^*)^3$ Row 4 $\Sigma(y-y^*)^4$
14	Ω Output address of inverse of augmented independent variables cross-products matrix.[4] SEQUENTIAL 1-15 and/or PRINT.
15	If weighting factors are desired, code this paramter 1 or -1 (see footnote 3). The weights must be in each row and must be to the right of the dependent variable (see Special Comments below). Weights should indicate a replication of an observation. Leave this parameter blank or code a zero if weights are not wanted.

Parameter Number	Use or Meaning
16	Tolerance used to determine if correlation matrix is singular. If this parameter is left blank, a tolerance of 10^{-5} will be used. If any other tolerance is desired, it should be punched as follows: *_.E-_* where the blanks could be filled in as follows: *13.5E-10*. This parameter must be enclosed in asterisks as shown in the examples.

IV. Special Comments

Sample Size, Regression Coefficients, Standard Errors, F Ratio, Multiple Correlations, T Ratio, and Dependent Variables Intercept are printed by default.

Thirty control variables will be allowed and are specified by normal conventions but these variables must be to the right of the dependent variables and weights. Control variables will not be in the calculations. If control breaks are used only the first set of output can be stored on Sequential address. (Control variables must be pre-sorted either in SOUPSORT or by Machine).

Independent variables must be on the left, then dependent variables, weights, (if any), and control variables (if any).

If the independent variable or the only dependent variable is constant a message will be printed and the sample will be discarded after computing the correlation matrix.

The index "0" on printed matrices refers to the intercept term.

V. Examples

(1)

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
MUL(C)(5)()()(P)()(P).
ENDS
DATA(7) (7F1.0)
.
.
.
.
.
.
END#
/*
```

This program reads from cards, uses 5 independent variables and 2 dependent variables. Means, Standard Deviations, Correlations, and default options are printed.

(2)

```
/*ID
// EXEC SOUP
//SYSIN  DD   *
TRA(C).
PER(1)(1,15)(20,22)(16,17)(19)(18)(23).
OUT(S1)(1,23).
PER(1)(1,15)(20,22)(16,17)(19)(18)(23).
ENDP
MUL(S1)(18)()(P)()()()(P)()()()()(P)(P)(1)*1.0E-8*.
$C-B (22)(23).
ENDS
DATA (23)(23F2.0)
  .
  .
  .
  .
  .
  .
  .
END#
/*
```

This program reads from Sequential 1, uses 18 independent variables, 2 dependent variables, weights, and control variables. Ouput are deviations, cross products, Durbin-Watson coefficients, sums of the second, third, and fourth powers of deviations, inverse of augmented independent variable cross products matrix and default options are printed.

VI. Footnotes

[1]Durbin-Watson Coefficients are a measure of autocorrelation with a distribution between 0 and 4.

[2]If weights are not used column 3 (sum of weights) is equal to the sample size (n).

[3]If weights (Parameter 15) are used two options are available. (1) Code Parameter 15 a -1 if the sum of weights (m) should be substituted for sample size in all calculations (warning--this may show a higher significance than is warranted from the data). (2) Code Parameter 15 a 1 if the sum of weights should be substituted only as follows:

$$\text{Mean: } \bar{X}_j = \frac{\sum_{i=1}^{n} w_i x_{ij}}{m}$$

$$\text{Covariance: } c_{jk} = \frac{X'X}{(n-1)} - \frac{(\Sigma wx_j)(\Sigma wx_k)}{m(n-1)}$$

[4]The dependent variable portion of the raw data cross-products matrix is deleted leaving the independent variable portion ΣX_i ΣX_j and n is then augmented.

$$A = \begin{array}{c|c} n & \Sigma x_k \\ \hline \Sigma x_j & X'X \end{array} \qquad \text{AND} \qquad A^{-1} = \begin{array}{c|c} W & Y \\ \hline Y' & Z \end{array}$$

such that for j=1, p and k=1, p

$$z_{jk} = \frac{r_{jk}^{-1}}{ns_j s_k}$$

$$Y = -\frac{(\Sigma x_j\ Z)}{n}$$

$$W = \frac{1 - (\Sigma x_k)Y}{n}$$

(from inversion by partitioning)

VII. Coefficients of Linear Dependency:

If the correlation matrix is singular a message will be printed as follows:

```
INPUT MATRIX IS SINGULAR.
THE FOLLOWING ROWS ARE LINEARLY DEPENDENT:

  1   2   .   .   .   .   .   N-1   N
```

Following this message the coefficients will be printed as follows:

```
COEFFICIENTS OF LINEAR DEPENDENCY

        N
  1   XXXX.XXXXX
  2   XXXX.XXXXX
  .
  .
  .
N-1   XXXX.XXXXX
```

The values are the unstandardized regression coefficients of variable N predicted by variables 1 through N-1. Those values which are approximately equal to zero are not part of the dependency.

PARTIAL CORRELATION

1. General Description

This routine, upon option, provides two of the more common types of special purpose correlation coefficients.

A. Partial Correlations:

This program produces coefficients of net correlation of any order from 1 to 19 in matrix form. Coefficients of successively higher order may be obtained by repeated calls to the program, each time using as input the previously generated partial correlation matrix; or several variables may be held constant at the same time by one call to the program.

The general equation used is:

$$r_{ij.abc...n} = \frac{r_{ij.abc...(n-1)} - r_{in.abd...(n-1)} \cdot r_{ij.abc...(n-1)}}{(1 - r^2_{in.abd...(n-1)})^{1/2} (1 - r_{jn.abc...(n-1)})^{1/2}}$$

References:

Mills, F.C. Statistical Methods, Holt, Rinehart and Winston. New York, 1955, 3rd edition.

B. Tetrachoric Correlations:

This type of correlation coefficient is used when continuous normally distributed variables are measured dichotomously.

This program is based on a program by Roald Buhler at Princeton University which in turn is based on a 650 program written at the Educational Testing Service. The approximation used was developed by Professor Ledyard Tucker.

2. Restrictions

A. Partial Correlations:

Input matrices may be no larger than 140 x 140 and must be compatible with SOUPAC conventions. In most cases the original input to the program will be a matrix of zero order correlations (see CORRELATION program write-up).

B. Tetrachoric Correlations:

This option is limited to 140 variables. All observations should be coded either 0 or 1. The program generates cross-count tables before computing the correlation coefficients.

. Parameters

The program name, PARTIAL CORRELATION, should be followed by the following parameters:

Parameter Number	Use or Meaning
1	Input address of R if partial correlations or raw data if tetrachoric correlations. (R is a correlation matrix).
2	Output Address of correlations desired.
3	0 if tetrachoric correlations are desired 1 if partial correlations are desired
4 - 22	Variables to be held constant in using partial correlations.

. Special Comments

When there is a zero cell or sufficiently close so that the tetrachoric correlation cannot be computed by this approximation, a value of -1. is used if the missing cell is off-diagonal. If a diagonal cell is zeroish (i.e., if a variable is all zero or all one) its correlations are set to 0.0.

Blanks are counted as zeroes.

. Examples

A series of observations of 8 variables are used to obtain zero order partial correlations with variables 5, 7, and 8 held constant:

```
/*JD
//  EXEC  SCUPAC
//SYSIN  DD  *
CORRELATIONS (CARDS)()(SEQ 1).
PARTIAL CORRELATION (SEQ 1) (PRINT)(1)(5)(7)(8).
END SCUPAC
DATA(8)(8F0.2)
   .
   .
   .
END *
```

REGRESSION-CORRELATION PROGRAM

I. Purpose

The purpose of this program is to run correlation and regression analysis. This program replaces the previous stand-alone programs Correlation, Canonical, Multiple Correlation, Step-Wise Multiple Correlation and Partial Correlation.

This program, through the use of the VARIABLE subparameter card, can be used to process subsets of the original input variables.

II. General Description

A. Correlation Section

w = weights
m = sum of weights $[m = \sum_{i=1}^{n} w_i]$
n = sample size (if Main Parameter 7 < 0, $n = m$)
x_{ij} = raw data; j^{th} variable; i^{th} observation
S = vector of Standard Deviation
$\bar{x}$ = means

Mean: $\bar{X}_j = \frac{\sum_{i=1}^{n} x_{ij}}{n}$ [if weights specified, $\bar{X}_j = \frac{\sum_{i=1}^{n} w_i x_{ij}}{m}$]

Cross-Products: $X'X = \sum_{i=1}^{n} (x_{ij} x_{ik})$ [if weights: $\sum_{i=1}^{n} (w_i x_{ij} x_{ik})$]

Covariance: $c_{jk} = \frac{X'X}{n-1} - \frac{(\Sigma x_j)(\Sigma x_k)}{n(n-1)}$

if weights (Main Parameter 7 > 0)

$$c_{jk} = \frac{X'X}{(n-1)} - \frac{(\Sigma wx_j)(\Sigma wx_k)}{m(n-1)}$$

Standard Deviation: $s_j = \sqrt{c_{jj}}$

Correlation: $r_{jk} = \frac{c_{jk}}{\sqrt{c_{jj} * c_{kk}}}$

B. Simple Linear Regression

To solve the equation $X_k = A_{jk} + B_{jk} X_j$

Regression Coefficients:

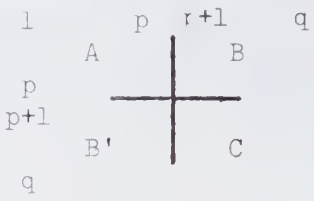
$$B_{jk} = r_{jk} \frac{S_j}{S_k}$$

Intercept: $A_{jk} = \bar{X}_k - B_{jk} \bar{X}_j$

C. Partial Correlations

p = number of variables to be partialed out
q = total number of variables

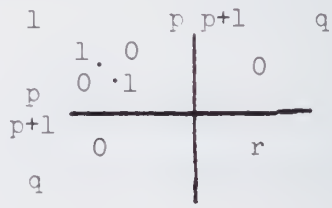
The correlation matrix is partitioned as follows:



$$D_{jk} = C - B'C^{-1}B$$

$$r_{jk \cdot 1 \ldots p} = \frac{D_{jk}}{\sqrt{D_{jj} * D_{kk}}}$$

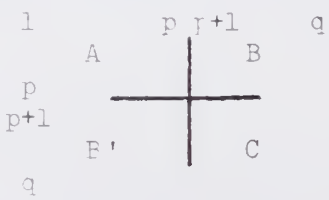
So output is of the form:



D. Canonical Correlations

The correlation matrix is partitioned as follows:

p = number of predictor variables
q = total number of variables



Standardized regression coefficients: $\beta = A^{-1}B$

R-Squared: $R^2 = B'A^{-1}B$

Eigenvalues of C = D Eigenvectors of C = H

Canonical Matrix: $(HD^{-1/2})'B'A^{-1}B \quad (HD^{-1/2})$

Eigenvalues-Canonical $R^2 = \lambda^2$ = Eigenvalues of Canonical Matrix

Canonical R-Correlation: λ (a diagonal matrix)

Wilk's Lambda: $\Lambda_j = \prod_{i=j}^{q-p} (1-\lambda^2_i)$ (for the j^{th} function)

Chi-Square: $\chi_j^2 = \log_e (\Lambda_j) (\frac{q+1}{2} - n)$

Eigenvectors of Canonical Matrix: V

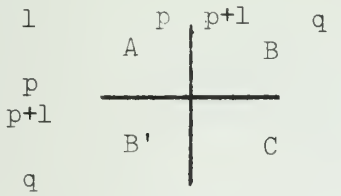
Criteria Weights: $W_c = (HD^{-1/2})V$

Predictor Weights: $W_p = \beta W_c \lambda^{-1}$

E. Multiple Correlation

p = number of independent variables
q = total number of variables

The correlation matrix is partitioned as follows:



Regression Coefficients (Standardized): $\beta = A^{-1}B$

Deviation Covariance: $D = S(C-B'\beta)S' \frac{n-1}{n-p-1}$ (unexplained variances)

Deviation (Partial) Correlations: $D_{jk} = D_{jk}/(D_{jj}D_{kk})^{1/2}$
(Correlations among dependent variables with the independent var. partialled out)

Regression Covariance: $C = SB'\beta S' \frac{n-1}{p}$ (explained variances)

Multiple Correlation: $R_j = (B'\beta)_j^{1/2}$

F Ratio: $F_{j\ n-p-1}^{\ p} = \left[\frac{R_j^2}{1-R_j^2}\right] \frac{n-p-1}{p}$ Testing Hypothesis $R^2 = 0$.

Standard Error of Estimate: $Se_j = s_j[(1-R_j^2) \frac{n-1}{n-p-1}]^{1/2}$

Regression Coefficients (Unstandardized): $b_{j.k} = \frac{s_j}{s_k} \beta_{j.k}$

Dependent Variable Intercept: $b_{o.k} = \bar{Y}_k - \sum_{j=1}^{p} b_{j.k} \bar{X}_j$

Standard Error of Regression Coefficients (Unstandardized):

$$s_{b_{j.k}} = S_{e_k}[(X'X)_{jj}^{-1}]^{1/2}$$

Standard Error of Regression Coefficients (Standardized):

$$S_{\beta_{j.k}} = \frac{S_k}{S_j} S_{b_{j.k}}$$

T = Regression Coefficient/Standard Error: $T_{j.k} = \frac{\beta_{j.k}}{s_{\beta_{j.k}}}$ Testing Hypothesis $\beta = 0$.

Predicted Dependent Variables: $y_k^* = b_{o.k} + \sum_{j=1}^{p} b_{j.k} x_j$

Deviations from Observed: $z_j = y_j - y_j^*$

Durbin-Watson Statistic: $d_j = \frac{\sum_{i=2}^{n} (z_{j_{i-1}} - z_{j_i})^2}{\sum_{i=1}^{n} (z_{j_i})^2}$

F. Stepwise Multiple Regression

In the step-wise procedure, intermediate results are used to give valuable statistical information at each step in the calculation. These intermediate answers are also used to control the method of calculation. A number of intermediate regression equations are obtained by adding one variable at a time thus giving the following intermediate equations:

a. $Y = B_0 + B_1 X_1$ where Y is the dependent variable

b. $Y = B_0 + B_1 X_1 + B_2 X_2$, etc.

The coefficients for each of these intermediate equations and the reliability of each coefficient are obtained by the step-wise procedure. The coefficients represent the best values when the equation is fitted by the variables included in the equation. The variable is added that makes the greatest improvement in "goodness of fit" or, stated another way, gives the greated reduction in variance of the dependent variable.

A variable may be indicated to be significant at an early stage and enter the regression equation. After several other variables are added to the regression equation, a variable in the equation may be indicated to be insignificant. Under this situaton the step-wise regression procedure will remove the insignificant variable before adding an additional variable. Thus, at the various steps in the regression procedure, only those variables which are significant will be included in the regression equation.

The F level to enter a variable controls when variables enter the equation and the F level to remove a variable likewise controls the removing of variables from the equation.

After the first step of regression subsequent coefficient and error terms depend on those which have gone before in an iterative manner.

For example, the standardized regression coefficients result from a partial inversion of the correlations matrix (replacing the correlations with the dependent variable). The diagonal elements of this inverse are also used. The multiple correlation in turn comes from the regression coefficients. As the iteration procedes with each step of regression new coefficients result.

Standardized regression coefficients: β_j $j = 1,\ldots,p$ where p is the number of independent var in the regression

Unstandardized regression coefficient: $B_j = \beta_i * \frac{S_Y}{S_j}$ Y is the dependent var.

Multiple correlation: $R = \sqrt{\sum_{j=1}^{P} r_{jY}\beta_j}$ r_{jY} is correlation of variable j with dependent variable

Intercept: $a = \bar{Y} - \Sigma B_j \bar{X}_j$

Standard error of mean of Y: $Se_{\bar{Y}} = S_Y \sqrt{1/(N-1)}$

Standard error of predicted Y: $Se_{\hat{Y}} = SY \sqrt{(1-R^2)/N-k-1)}$ $\hat{Y}$ is predicted Y

Standard error of estimate: $Se_{est} = SY \sqrt{(1-R^2)(N-1)/N-k-1)}$

$= Se_{\hat{Y}} \sqrt{N-1}$

Standard error of unstandardized coefficient: $Se_{B_j} = (Se_{\hat{Y}}/S_j) \sqrt{D_j}$
D_j is diagonal element of partially inverted correlation

Standard error of standardized coefficient: $Se_{\beta_j} = Se_{B_j} * \frac{S_j}{S_Y}$

T ratio: $T = \frac{B_j}{Se_{B_j}}$ Degrees of freedom: $Df = N-p-1$

The above are printed by default at the end of each iteration.

III. Parameters

A. Main Parameter Card

REG		Mnemonic
1		Input Address of raw data (default), Augmented Cross Products, Covariance or Correlations Matrices (see notes on Type of Input).
2		Output Address of Means, Standard Deviation, Sample Size, Sum of Weights (stored in 4 rows).
3	Ω	Output Address of Correlations (if subparameters are specified, the correlation matrix must be saved on a sequential file).
4	Ω	Output Address of Covariance Matrix.
5	Ω	Output Address of Raw Data Cross Products Matrix.
6		Input Address of labels.
7		Weights (1, 0, -1; Default 0)
8		TYPE of Inputs (see notes on types of input) 0 Raw Data (default) 1 Cross Products 2 Covariance 3 Correlation 4 Correlation
9		Search for Pivotal Elements 0 or Default - no search 1 Perform search over entire matrix
10		Test for Positive Definiteness 0 or Default - perform test 1 Ignore test
11		Tolerance (Default *1.0E-5*), must be enclosed in asterisks, "* *".

(9, 10, 11: for all inversions)

12		Output Address of Augmented Correlation matrix suitable for input (print not allowed, correlation matrix must have been saved on a sequential file).

B. Subparameter Card

1. CONTROL (mnemonic: CON)

CON	(must be first subparameter card, if used)
1-20	Control Variables (Control Variables must be eliminated for any further analysis. Only the first subset is output to a sequential file.)

2. VARIABLE (mnemonic: VAR)

Purpose: To specify a subset and order of variables to be used in all following subparameter cards until another VAR card is encountered.

The absence of any VAR cards or a VAR card without parameters indicates that all the input variables are to be used in their original order.

The parameters always refer to the order of variables originally entered.

Form: Index sets (see below).

Examples:

1. VAR(1)(2)(3)(5)(7).
   Variables 1, 2, 3, 5, and 7 are used (5 variables).
2. VAR(1,2)(3,7).
   Variables 1 through 2 and 3 through 7 are used (7 variables).
3. VAR(1,2)(3,7,2).
   Variables 1 through 2 and 3 through 7 in steps of 2 (3, 5, and 7) are used (5 variables).

Caution: If a variable is specified more than once, the subparameter cards will use the variable more than once. This may cause singular matrices in MUL or CAN.

3. SIMPLE LINEAR REGRESSION (Mnemonic: SIM)

$$X_k = B_{jk}X_j + A_{jk}$$

1	Ω	Output Address of Coefficients (B).
2	Ω	Output Address of Intercepts (A).

4. PARTIAL CORRELATIONS (Mnemonic: PAR)

The first N variables are partialled out. The variables may be reordered by use of a VAR card. The correlations with those variables which are partialled out are set to 0, the diagonal of the matrix is set to 1.

1		Number of variables to be partialled out.
2	Ω	Output Address of Partial Correlation Matrix.

5. MULTIPLE LINEAR REGRESSION (Mnemonic: MUL)

1		Number of independent variables. Ind. var. + dep. var. + wt. var. + control var. ≤ 450. Ind. var. + dep. var. ≤ 430.

2	Ω	Output Address of coefficients (unstandardized). SEQUENTIAL 1-15, PRINT is default. Coefficients for P independent and L dependent variables are written as L rows with L + P + 1 columns each. The i^{th} row contains in order the i^{th} intercept term, the P coefficients for the i^{th} dependent variable, a -1 in the P + i + 1 location, and 0's for all other locations. This format is compatible with the ECONOMETRICS REDUCED FORM AND RESIDUAL ANALYSIS program.
3	Ω	Output address of inverse of augmented independent variables cross-products matrix.[4] SEQUENTIAL 1-15 and/or PRINT.
4		Output Address of predicted dependent variables. SEQUENTIAL 1-15 and/or PRINT (also P(F) is optional).
5		Output address of deviations from actual. SEQUENTIAL 1-15 and/or PRINT (also P(F) is optional).
6	Ω	Output Address of Durbin-Watson and second, third and fourth powers of sums of deviations. SEQUENTIAL 1-15 and/or PRINT. Row 1 Durbin-Watson Coefficients. Row 2 $\Sigma(y-y^*)^2$ Row 3 $\Sigma(y-y^*)^3$ Row 4 $\Sigma(y-y^*)^4$
7	Ω	Output Address of deviation covariance matrix. SEQUENTIAL 1-15 and/or PRINT.
8	Ω	Output Address of deviation (partial correlation matrix. SEQUENTIAL 1-15 and/or PRINT.
9	Ω	Output Address for regression covariance matrix. SEQUENTIAL 1-15 and/or PRINT.

Raw data must be stored on a sequential file.

6. CANONICAL CORRELATIONS (Mnemoic: CAN)

The predictor variables must precede the criteria variables. Reordering may be done by use of a VAR card.

1		Number of predictors (must be greater than the number of criteria).
2	Ω	Output Address of Predictor Weighting Matrix.
3	Ω	Output Address of Criteria Weighting Matrix.
4		Output Address of Standardized Regression Coefficients (Print only).
5		Output Address of R^2 (Print only).

6	Ω	Output Address of Standardized Predictor Weighting Matrix.*
7	Ω	Output Address of Standardized Criteria Weighting Matrix*.
8		Output Address of Eigenvalues (Sequentials only) as a row.

* Weighting matrices are standardized so that the sum of the square for each function (column) is equal to 1.

7. STEPWISE MULTIPLE REGRESSION

Only one dependent variable is allowed. It must be the rightmost variable.

1	"F" level to enter an independent variable into the regression equation. An example would be: *4.0*.
2	"F" level to remove a variable from the regression equation. An example would be: *4.0*.
3	1 if constant term in equation is assumed to equal zero (0). [If this option is used the raw data cross-products matrix must be saved on a Sequential file.]
4	Output Address of coefficients (Sequential file, only).
5	Output Address of predicted dependent variables (Print only). Input may not come from cards.
6	First (N) variables are placed in regression first.
7	First (N) variables are to be kept in the regression once they are entered.
8	1 if intermediate steps of regression are not to be printed.

V. Explanation

A. Type of Input (Main Parameter 8)

0	RAW DATA is input

TYPE

1 — An augmented Cross Products Matrix is input. The form is as follows:

n	ΣX
ΣX	$X'X$

2 — An augmented Covariance Matrix is input. The form is as follows:

n	ΣX
ΣX	C_{jk} (covariance)

3 — An augmented Correlation Matrix is input. The form is as follows:

n	$\bar{X}_k$ (means)
(Std. dev.) S_j	r_{jk} (correlations)

This option may be used to input a previously calculated matrix (Correlation, Covariance, or Cross-Products) for use as input to a sub-operation.

B. Weights (Main Parameter 7)

If the Weights Flag is $\neq 0$ and raw data is input, then the weighting variable must be the rightmost variable.

If the Weights Flag is > 0 then the input matrices of type 1, 2, or 3 should be augmented as follows:

TYPE — FORM

1

n	ΣX
ΣX	$X'X$
m	---

2

n	ΣX
ΣX	C_{jk} (covariance)
m	---

3

n	$\bar{X}_k$
s_j	r_{jk}
m	---

If the Weights Flag is < 0 then the sum of weights should be substituted for n.

C. Miscellaneous

1. If suboperations are requested, the Correlation matrix must be output to a Sequential file. Be sure not to use this file as an output address for any subparameter cards.

2. All output is in the form of a matrix permuted in order of the variable specified on a VAR card.

3. All printed matrices are given variable numbers determined upon entry at the main input address.

4. If Predicted dependent variables and/or deviation from predicted values in MULTIPLE Regression are printed, the use of labels will increase the number of lines printed by 20% to 25%.

V. Examples

A. Labels

The label deck must precede any data input to this program by cards. Up to 8 characters may be used to label each variable, therefore the format field may be up to 8 columns in width. The labels should be left justified within each field.

```
REG(C)()(S1)()()(C).
  .
  .
  .
ENDP
ENDS
DATA(7)(7A8)
ONE     TWO     THREE     FOUR     FIVE     SIX     SEVEN
END#
DATA(7)(7F2.0)
  .
  .
  .
END#
```

B. VARIABLE Card

1. MULTIPLE CORRELATIONS

To run Multiple Correlation on the following equations:

#1 $X_5 = b_0 + b_1X_1 + b_3X_3$

#2 $X_1 = b_0 + b_2X_2 + b_4X_4 + b_5X_5 + b_6X_6$

```
REG(C)(P)(S1/P).
VAR(1)(3)(5).
MUL(2).          #1
VAR(2,6,2)(5)(1).
MUL(4)           #2
ENDP
```

2. CANONICAL Correlations

a. To permute the predictor and criteria variables.

```
REG(C)()(S1).
VAR(7,14)(1,6).
CAN(8)(P)(P)(P)(P).
ENDP
```

b. To use different subsets of predictor or criteria variables.

$\sum_i b_i X_i = \sum_j b_j X_j$

#1 i = 7, 8, 9, 14; j = 1, 2, 3

#2 i = 7, 8, 9, 10, 11; j = 1, 2, 4, 5, 6

```
REG(C)()(S1).
VAR(7,9)(14)(1,3).
CAN(4)(P)(P)(P)(P).     #1
VAR(7,11)(1,2)(4,6).
CAN(5)(P)(P)(P)(P).     #2
ENDP
```

3. PARTIAL Correlations

To specify the order of variables where the first n variables will be partialled out.

#1 $r_{ij.789}$

#2 $r_{ij.1578}$ (assume 10 variables)

```
REG(C)()(S1).
VAR(7,9)(1,6)(10).
PAR(3)(P).
VAR(1,5,4)(7,8)(2,4)(6)(9,10).
PAR(4)(P).
ENDP
```

C. Input Types

1. Augmented Cross Products TYPE=1

```
MAT.
GEN*S2*1*.
MOV(C)(S1).   RAW DATA
EXP(S2)(S1)(S3).
HOR(S3)(S1)(S2).
TRA(S2)(S3).
MUL(S3)(S2)(S1).
ENDP
REG(S1)(P)(S2/P)()()()()(1).
  .
  .
  .
ENDP
ENDS
```

2. Augmented Correlation TYPE=3

```
REG(C)()(S1)()()()()()()()()(S2).
ENDP
REG(S2)()(S1)()()()()(3).
  .
  .
  .
ENDP
ENDS
```

3. Unaugmented Correlation TYPE=4

```
COR(C)()(S1).
REG(S1)()(S2)()()()()(4).
  .
  .
ENDP
```

D. CONTROL Card

The CONTROL Card is similar in use to a $C-B card with the CORRLEATIONS program. The following forms are equivalent.

```
REG(C)(P)(S1/P).
CON(5)(9)(11).
ENDP

COR(C)(P)(S1/P).
$C-B(5)(9)(11).
```

E. Equivalent Forms

This section indicates the form needed to replace current stand-alone programs with the Regression Program.

The superscripts indicate like uses of the parameters between the two forms.

1. Correlations

   a. $COR(C^{1})(P^{2})(P^{3})(P^{4})(P^{5})(P^{6})(P^{7})$.

      $REG(C^{1})(P^{2})(S1/P^{3})(P^{5})(P^{4})$.
      $SIM(P^{6})(P^{7})$.
      ENDP

   b. $COR(C^{1})(P^{2})(P^{3})$.

      $REG(C^{1})(P^{2})(P^{3})$.
      ENDP

2. Multiple Correlations

   a. $MUL(S1^{1})(6^{2})(P^{3})(P^{4})(P^{5})(P^{6})(P^{7})(P^{8})(P^{9})(P^{10})(P^{11})(P^{12})(P^{13})(P^{14})$.

      $REG(S1^{1})(P^{5})(S2/P^{7})(P^{9})(P^{8})$.
      $MUL(6^{2})(P^{6})(P^{14})(P^{3})(P^{4})(P^{13})(P^{10})(P^{11})(P^{12})$.
      ENDP

   b. $MUL(C^{1})(5^{2})()^{3}()^{4}(P^{5})()^{6}(P^{7})$.

      $REG(C^{1})(P^{5})(S1/P^{7})$.
      $MUL(5^{2})$.
      ENDP

3. Canonical Correlations

   a. $COR(C^{1})()(S1)$.
      $CAN(S1)(50^{2})(P^{3})(P^{4})(8^{5})(6^{6})(1^{7})(1^{8})(1^{9})$.

```
          1
REG(C)()(S1).
     5   3   4   8   9
CAN(8)(P)(P)(P)(P).
ENDP
```

b.
```
     1  2   3   4   5   6   7   8   9
CAN(C)()(P)(P)(8)(6)(1)(1)(1).
```

```
          1
REG(C)()(S1)()()()()(4).
     5   3   4   8   9
CAN(8)(P)(P)(P)(P).
ENDP
```

4. Partial Correlations

a.
```
          1
COR(C)()(S1).
          2   3   4   5   6
PAR(S1)(P)(1)(1)(2)(3).
```

```
          1
REG(C)()(S1).
          2
PAR(3)(P).
ENDP
```

b.
```
     1   2   3   4   5   6
PAR(C)(P)(1)(1)(2)(3).
```

```
          1
REG(C)()(S1)()()()()(4).
          2
PAR(3)(P).
ENDP
```

Addendum: The following additional subparameter card is available.

TRANSFORMATIONS (Mnemonic: R-T)

A matrix of transformations of the correlation matrix is output to a specified address. The Z-transformed correlations will be approximately normally distributed, with a mean of $\tanh^{-1} \rho$, where ρ is the population correlation coefficient, and variance $1/(n-3)$.

The T-transformed correlations will have the students t-distribution with n-2 degrees of freedom.

1 Output address of fisher Z transformations.

2 Output address of T-transformations.

STEP-WISE MULTIPLE CORRELATION

I. General Description

The STEP-WISE MULTIPLE CORRELATION program calculates the following basic statistics before the step-wise procedure begins. All variables are included.

Mean: $\bar{X}_i = \frac{\Sigma X_i}{N}$ Crossproducts: $P_{ij} = \Sigma(X_i X_j)$

Covariance: $S_{ij} = \frac{N\Sigma(X_i X_j) - (\Sigma X_i)(\Sigma X_j)}{N(N-1)}$ All summation is over the sample.

Standard Deviation: $S_i = (S_{ii})^{1/2}$

Product Moment Correlation: $r_{ij} = \frac{S_{ij}}{S_i S_j}$

In the step-wise procedure, intermediate results are used to give valuable statistical information at each step in the calculation. These intermediate answers are also used to control the method of calculation. A number of intermediate regression equations are obtained by adding one variable at a time thus giving the following intermediate equations.

a. $Y = B_0 + B_1 X_1$ where Y is the dependent variable.

b. $Y = B_0 + B_1 X_1 + B_2 X_2$, etc.

The coefficients for each of these intermediate equations and the reliability of each coefficient are obtained by the step-wise procedure. The values and reliability may vary with each subsequent equation. The coefficients represent the best values when the equation is fitted by the variables included in the equation. The variable is added that makes the greatest improvement in "goodness of fit" or, stated another way, gives the greatest reduction in variance of the dependent variable.

A variable may be indicated to be significant at an early stage and enter the regression equation. After several other variables are added to the regression equation, a variable in the equation may be indicated to be insignificant. Under this situation the step-wise regression procedure will remove the insignificant variable before adding an additional variable. Thus, at the various steps in the regression procedure, only those variables which are significant will be included in the regression equation.

The F level to enter a variable controls when variables enter the equation and the F level to remove a variable likewise controls the removing of variables from the equation.

The last step in the step-wise procedure predicts the value of the dependent variable for each set of observations based on the final regression equation. Deviation between the actual and predicted values are also calculated.

(See parameter 4).

II. Calculations and Formulas

After the simple correlations and the first step of regression subsequent coefficients and error terms depend on those which have gone before in an iterative manner.

For example, the standardized regression coefficients result from a partial inversion of the correlations matrix (replacing the correlations with the dependent variable). The diagonal elements of this inverse are also used. The multiple correlation in turn comes from the regression coefficients. As the iteration procedes with each step of regression new coefficients result.

Standardized regression coefficients: β_i $i = 1,\ldots,k$ where k is the no. of indep. var in the regression

Unstandardized regression coefficient: $B_i = \beta_i \frac{S_Y}{S_i}$ Y is the dependent variable

Multiple correlation: $R = \sqrt{\sum_{i=1}^{k} r_{iY}\beta_i}$ r_{iY} is correlation of variable i with dep. var.

Intercept: $C = \bar{Y} - \Sigma\beta_i\bar{X}_i$

Standard error of mean of Y: $Se_Y = S_Y\sqrt{1/(N-1)}$ N is sample size

Standard error of predicted Y: $Se_{\hat{Y}} = S_Y\sqrt{(1-R^2)/N-k-1)}$ $\hat{Y}$ is predicted Y

Standard error of estimate: $Se_{est} = S_Y\sqrt{(1-R^2)(N-1)/N-k-1)}$

$= Se_{\hat{Y}}\sqrt{N-1}$

Standard error of unstandardized coefficient: $Sc_{B_i} = (Se_{\hat{Y}}/Si)\sqrt{Di}$ Di is diagonal element of partially inverted correlation matrix

Standard error of standardized coefficient: $Se_{\beta_i} = Se_{B_i}\frac{S_i}{S_Y}$

T ratio: $T = \frac{\beta_i}{Se_{\beta_i}}$ Degrees of freedom: $Df = N-k-1$

III. References

A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, New York, Wiley and Sons, 1960, pp. 191-195.

IV. Restrictions

The dependent variable must be after all the independent variables in each row. Only one dependent variable is allowed per program call.

The input data to this program may come from any source conforming to SOUPAC. Output may be PRINT only except for coefficients (see parameters).

V. Output

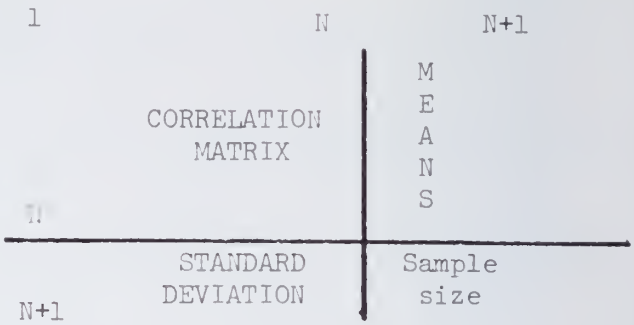
The following is normally printed: Crossproducts, Means and Standard Deviations, Covariance, Correlations, Standard Error of Mean of Dependent Variable. Plus for each step of regression: F level, Standard Error of Predicted Dependent Variable, Multiple Correlation, Standard Error of Estimate, Dependent Variable Intercept, Degrees of Freedom for F. And, for each Independent Variable in the step: Unstandardized Regression Coefficient, Standard Error of Unstandardized Regression Coefficient, Standardized Regression Coefficient, Standard Error of Standardized Regression Coefficient, T, Degrees of Freedom for T. All printing may be suppressed except the final step of regression.

The coefficients which may be requested on temporary storage are unstandardized. The output row consists of the intercept, the coefficients, and a minus 1. Any independent variable not entered into regression will get a zero coefficient output.

VI. Parameters

The parameters for the STEP-WISE MULTIPLE CORRELATION program appear on the program call card. They must follow the program name in this order:

Parameter Number	Descriptions
1	Input Address. CARDS or SEQUENTIAL 1-15. (see parameter 4 for special conditions.)
2	"F" level to enter independent variable into the regression equation. An example would be: *4.0*
3	"F" level to remove a variable from the regression equation. An example would be: *4.0*.
4	This parameter should be set to 1 if the predicted dependent variables are to be calculated. (If this option is needed, input data must not be from card.) 0 or blank if not wanted.
5	1 if constant term in equation is assumed to equal zero (0).
6	1 if want to use weighting factor. (If a weighting factor is used, it must be the last variable in the input data row.)

Parameter Number	Description
7	1 if intermediate steps of regression are not to be printed.
8	1 if do not want cross-product matrix printed; 2 if input data is in the following form:  (where N is the number of variables
9	1 if do not want means and standard deviations printed
10	1 if do not want covariance to be printed
11	1 if do not want correlations to be printed.
12	Tolerance to be used to determine when singularities are assumed to occur. If this parameter is left blank 10^{-5} is used. If it is desired to change this parameter, the following would be used: *1.E-10* where any number could be substituted for the 10.
13	Output (intermediate storage) of coefficients
14	First (N) variables are placed in regression first.
15	First (N) variables are kept in regression, if entered.

VII. Special Comments

The dependent variable must be the last variable in the input row (unless a weighting factor is used, then the dependent variable will be the next to the last variable in the input row.)

DISTRIBUTION ANALYSIS PACKAGE

FIT
(CHI-SQUARE GOODNESS-OF-FIT TEST)

I. General Description

In statistical applications, it is frequently the case that certain assumptions were made concerning the probability distribution of a random variable. A frequent assumption is that a particular variable is normally distributed. The question that arises is "how valid an assumption was this?" A method of testing this assumption (or hypothesis) is the CHI-SQUARE GOODNESS-OF-FIT TEST.

This program provides tests of hypothesis that user's data is (1) a random sample from the distribution $P\theta$, where θ is a user specified parameter, or (2) a random sample from a class, $P\theta$, of distributions where θ is not specified. These will be called tests of type 1 and type 2, respectively.

Distributions which can currently be tested in the program are the binomial, Poisson, normal, gamma and continuous rectangular distributions. The user may, for example, wish to test the hypothesis that his data was from a normal distribution with variance 1, for some mean.

The program on the basis of the user provided information decides on a set of points $X_1 < X_2 < \ldots . X_n$ in the range of the distribution under consideration. The test then compares the observed numbers, o_i, of observations in each interval $[X_{i-1}, X_i)$, with the expected numbers

$$e_i = P_{\hat{\theta}} \{X_{i-1} \leq Y_1 \leq X_i\} \quad , \quad \{Y_i \; ; \; 1 \leq i \leq m \}$$

Where $\hat{\theta}$ is the user specified value of θ, if it was specified, and if not, $\hat{\theta}$ is the maximum likelihood estimator for a hypothesis of form (2) above. The comparison of o_i with e_i is made by the statistic

$$X^2 = \sum_{i=1}^{n+1} \frac{(o_i - e_i)^2}{e_i} \quad .$$

The distribution of this statistic has an approximate X^2 distribution when the hypothesis is true and n is large. (See Billingsley, 1961). The program prints the computed value of X^2 and the number of degrees of freedom for the test.

If one sample, with m observations, is to be tested, input to the program from cards or sequential will be one variable with m observations. More than one sample may be tested at a time, if the same distribution, or in some cases a distribution from the same class, is the one being considered. Thus a test of each of three samples, all of which are on the same sequential storage or card deck, may be performed.

II. Distributions Available

NOTATION: In the description below f(x) denotes the value of the probability density function of the point x. F(x) is the cumulative distribution function defined as $\Pr\{X \leq x\}$

A. DISCREET DISTRIBUTIONS

The Distributions are classified here as being "discreet," (i.e. having positive probability at a countable number of values of the random variable it describes) or as "Continuous" (having positive density over a continuous range of values of the random variable).

1. BINOMIAL(N,P) N a positive integer; $0 \leq p \leq 1$

$$f(x) = \binom{N}{x} p^x (1-p)^{N-x} \qquad \text{for } x \text{ an integer such that } 0 \leq x \leq N$$

$$F(x) = \sum_{i=0}^{x} \binom{N}{i} p^i (1-p)^{N-i} \qquad \text{for } x \text{ an integer such that } 0 \leq x \leq N$$

This distribution is appropriate, for example, in situations where a random variable is sampled independently (observed) N times, the observations, and where the values of the random variable can be classified precisely as being in one of two sets often denoted "success," and "failure," with probabilities p, and q = 1 - p respectively.

2. POISSON (λ) $\lambda > 0$

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!} \qquad x \geq 0 \text{ ; and } x \text{ an integer}$$

$$F(x) = \sum_{i=0}^{x} \frac{\lambda^i e^{-\lambda}}{i!} \qquad x \geq 0$$

The Poisson distribution is often used as an approximation to the binomial when the number of observations is large, p = Pr {success} is small, and the product n * p is essentially a constant. The Poisson is a pervasive distribution in its own right, arising in situations where the probability of no occurrences, or of one occurrence of a phenomenon in a unit of time is moderate, where the probability of more than one occurrence is essentially negligible in comparison, and where the frequencies of occurrence in adjacent intervals are independent of each other.

B. CONTINUOUS DISTRIBUTIONS

1. NORMAL (μ, σ^2) $\quad -\infty < \mu < +\infty \quad ; \sigma^2 > 0$

$$f(x) = \frac{1}{\sqrt{2\Pi\ \sigma^2}} \quad \exp \frac{-(X - \mu^2)}{2\ \sigma^2} \qquad -\infty < x < +\infty$$

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\ \Pi\ \sigma^2}} \quad \exp \frac{-(t - \mu)^2}{2\sigma^2} \quad dt \qquad -\infty < x < +\infty$$

Through the Central Limit Theorem, the use of this distribution has been justified to describe a tremendous variety of phenomena in which the random variable under consideration is assumed to be the sum of a large number of independent random variables, each having a small contribution to the total.

2. GAMMA (α,β) $\quad \alpha > 0 \quad , \quad \beta > 0$

$$f(x) = \frac{\beta^{\alpha}\ x^{\alpha-1}\ e^{-\beta x}}{\Lambda(\alpha)} \qquad \text{for } x > 0$$

$$F(x) = \int_{0}^{x} \frac{\beta^{\alpha}\ t^{\alpha-1}\ e^{-\beta t}\ dt}{\Lambda(\alpha)}$$

Here $\Lambda(\alpha) = (\alpha-1)\,\Lambda(\alpha-1)$ and so if α is a non-negative integer, $\Lambda(\alpha) = (\alpha-1)!$

The gamma distribution is the sampling distribution for the sum of α independent identically distributed "negative exponential" random variables, with $\beta = \lambda$ where λ is the parameter for the negative exponential distribution. The negative exponential distribution is itself the special case gamma $(1, \beta)$. The gamma distribution is used in dealing with waiting times, where the expected frequency in a given interval has Poisson distribution. The "Chi-Square" distribution is another important special case of the gamma distribution where

$$\chi^2\ (r) = \text{gamma}\ (r/2,\ 1/2).$$

3. RECTANGULAR (Θ_1, Θ_2) $\qquad \Theta_1 < \Theta_2$

$$f(x) = \frac{1}{(\Theta_2 - \Theta_1)} \qquad \Theta_1 \le x \le \Theta_2$$

$$F(x) = \int_{\Theta_1}^{x} \frac{1}{\Theta_2 - \Theta_1}\ dt = \frac{x-\Theta_1}{\Theta_2-\Theta_1} \qquad \Theta_1 \le x \le \Theta_2$$

This distribution is appropriate where any event in an interval $\{\theta_1, \theta_2\}$ has equal probability of occurring, but no occurrences will be outside the interval.

III. Use of the Program

A. PRELIMINARY

The program is invoked by the main parameter card (a SOUPAC "Program Card"). Information concerning the distribution and intervals to be used are supplied on subparameter cards for the program.

Input is in the form of column vectors, and may be from cards or sequential storage. Each variable (vector of data will be tested against the same type of distribution, i.e. information provided on the distribution card applies to all variables.

The subparameter card provides the means of supplying information to the program concerning the distribution to be used, as well as the number and size of intervals to be used in the tests. In some cases a distribution parameter must be supplied by the user.

B. MAIN PARAMETER CARD

Immediately following the program name CHI-SQUARE GOODNESS-OF-FIT (mnemonic: FIT), the following parameters are listed.

Parameter Number	Use of Meaning
1	Input Address. May be CARDS or SEQUENTIAL (S1, or S2, etc).

C. SUBPARAMETER CARDS

1. GENERAL DESCRIPTION

One "distribution card," chosen from the list below, should appear with each call to the program. For tests of type 1 ("simple hypothesis") the parameters of the distribution are specified on this card. For distributions allowing tests of type 2 ("composite hypothesis") one or more of these parameters may be left blank. Other parameters on the card relate to intervals to be used for determining observed and expected frequencies. Because this test is "asymptotically valid," these should be chosen so that expected frequencies of any interval does not fall below 5.

2. DISTRIBUTION CARDS

Where * * is used, a parameter requires a decimal point number; when ( ) appears an integer is required.

DISTRIBUTION

BINOMIAL(M)*P*(N).

M and P are parameters of the distribution, P the probability of success on a trial, M the number of trials. N is the number of points (integral values starting at 0) to be grouped in each interval. For a test of Type 2, the parameter P should be left blank.

POISSON*λ*(ENDPOINT)(N).

λ is the Poisson density parameter, and may be left blank for a test of type 2. N is the number of adjacent points to be grouped in each interval until ENDPOINT is reached. The Interval [ENDPOINT, +∞) is then the last interval.

RECTANGULAR*Θ1**Θ2*(N).

Θ1, and Θ2 are the endpoints for the range of the distribution Θ1 < Θ2. N is the number of equal sized intervals to use for the test. Only tests of type 1 are allowed with this distribution.

NORMAL*μ**σ^2**STPT**ΣP*(N).

μ is the mean of the distribution, σ^2 is the variance. Either or both of the parameters may be left blank for a test of type 2. STPT and ΣP are the start point and end point of an interval to be broken up into N equal sized intervals for the test. The additional intervals (-∞, STPT), and [ΣP, +∞) will be used.

GAMMA*α**β**ENDPOINT*(N).

α is the degree's of freedom parameter, and must be specified. *β* is the density parameter and may either be specified for a type 1 test or left out for a type 2 test. The portion of the real line between 0 and ENDPOINT will be divided into N equal sized intervals for the test. One additional interval, [ENDPT, +∞) will be included.

IV. Restrictions

A maximum of 50 input variables (each variable a sample to be tested) may be input to the program. Only one distribution or distribution type (e.g. NORMAL with variance 1 and any mean) may be tested.

V. Examples

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
FIT(CARDS).
NOR*2.0****-10**14*(24).
END P
END S
DATA(1000,1)(21X,F4.1)
.
.
.
.
#END
/*
```

This program will test the hypothesis that the sample is from a normal distribution with a mean of 2, for some variance. The interval [-10,14) will be divided into 24 pieces, each of length 1, for the test.

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
FIT(CARDS).
BIN(10)*1/2*(2).
END P
END S
DATA(600,1)(26X,F2.0)
.
.
.
.
#END
/*
```

This program will test the simple (type 1) hypothesis that the sample was from a B (10,1/2) population. Note that with the given number (600) of observations we had to group adjacent pairs of points into the same interval to insure a reasonable expected frequency in every cell.

THE KOLMOGOROV-SMIRNOV STATISTIC

I. General Description

The program computes the Kolmogorov-Smirnov (K-S) D statistic,

$$D = \sup_{\text{all } x} \left| F_N(x) - F(x) \right| ,$$

where F_N is the sample cumulative distribution for a sample of size N and $F(x)$ is the specified cumulative distribution.

II. Theoretical Discussion

The empirical distribution function which is determined by the order sample $x_{(1)} \cdots x_{(n)}$; =

$$F_N(x) = \begin{cases} 0 \text{ for } x < x_{(1)} \\ j/n \text{ for } x_{(j)} \leq x < x_{(j+1)} \\ 1 \text{ for } x \geq x_{(N)} \end{cases}$$

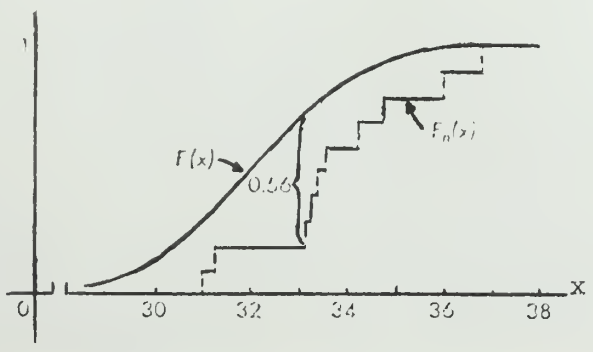
will generally differ from the population distribution function. If the sample distribution differs exceedingly from the specified distribution $F(x)$, the amount of the difference might be of use in determining whether to accept the hypothesized distribution as correct. The Kolmogorov-Smirnov test uses the maxmimum actual numerical difference $\left| F_N(x) - F(x) \right|$.

Example (See Lindgren [4]):

Consider testing the hypothesis that a distribution is normal with mean = 32 and variance = 3.24 with 10 sample observations

31.0	33.7
31.4	34.4
33.3	34.9
33.4	36.2
33.5	37.0

$F_N(x)$ and $F(x)$ are sketched below.



D = .56 which is the maximum of $|F_N(x) - F(x)|$.

At the .95 confidence level the critical value D' = .40925

Since D > D' the distribution being tested is rejected at the 5% level.

* See Owen [5]

III. Notes

Only supply the parameters needed to determine a specified distribution. This program, at this time, calculates the Kolmogorov-Smirnov statistic for the following distributions if given the proper parameters.

A. Normal
   1. mean
   2. variance
B. Central Chi-Square
C. Noncentral Chi-Square
   1. degrees of freedom
   2. noncentrality parameters*
D. Central F
   1. degrees of freedom numerator
   2. degrees of freedom denominator
E. Noncentral F
   1. degrees of freedom numerator
   2. degrees of freedom denominator
   3. noncentrality parameter*
F. Central Beta
   1. degrees of freedom numerator
   2. degrees of freedom denominator
G. Noncentral Beta
   1. degrees of freedom numerator
   2. degrees of freedom denominator
   3. noncentrality parameter*
H. Student's t
   1. degrees of freedom
I. Gamma
   1. A
   2. B

   where $F(x) = \int_0^x \frac{1}{\Gamma(A)B^A} t^{(A-1)} e^{-(A/B)} dt$

J. Exponential - special case of Gamma
   1. A = 1
   2. B = 1/L where L = rate of occurence
K. Noncentral T - transform to noncentral F
   1. degreesoof freedom numerator
   2. degrees of freedom denominator
   3. noncentrality parameter*

* Noncentrality parameter is defined: $\lambda = \frac{1}{2} \sum_{i=1}^{N} \mu^2$
(See Graybill [3] for further development of λ).

IV. Sample Programs

A. Normal - type 1
μ = 32
σ = 3.24

```
K-S(C)(1)  *32**3.24*.
```

B. Central Chi-Square - type 2
d.f. = 24

```
K-S(C)(2)****(24).
```

C. Noncentral Chi-Square - type 2
d.f. = 10
λ = .25

```
K-S(C)(2)****(10)()(.25).
```

D. Central F - type 3
d.f. numerator = 5
d.f. denominator = 16

```
K-S(S1)(3)****(5)(16).
```

E. Noncentral F - type 3
d.f. numerator = 10
d.f. denominator = 13
noncentrality parameter (λ) = 2.5

```
K-S(S2)(3)****(10)(13)*2.5*.
```

F. Central Beta - type 4
d.f. numerator = 5
d.f. denominator = 25

```
K-S(S2)(4)****(5)(25).
```

G. Noncentral Beta - type 4
d.f. numerator = 8
d.f. denominator = 20
noncentrality parameter = 2.8

```
K-S(S1)(4)****(8)(20)*2.8*.
```

H. Student's t - type 5
d.f. = 23

```
K-S(C)(5)****(23).
```

I. Gamma - type 6
A = 10
B = 3

```
K-S(S3)(6)*10**3*.
```

J. Exponential - type 6
A = 1
B = 1/.5 = 2 where L = .5

```
K-S(S2)(6)*1**2*.
```

K. Noncentral t - type 3
A noncentral t distribution can be transformed into a non-central F distribution by squaring the values.
Degrees of freedom = 5
Noncentrality parameter = .25

```
TRA(C).
MUL(1)(1)(1).
OUT(S1)(1).
END P
K-S(S1)(3)****(1)(5)*.25*.
```

V. Parameters

The following parameters follow the mnemonic K-S:

Parameter Number	Description
1	Input address, CARDS, SEQUENTIAL 1-15.
2	Specified distribution (see Section IV).
3	Floating point value (Mean, A).
4	Floating point value (Variance, B).
5	Degrees of freedom (numerator).
6	Degrees of freedom (denominator).
7	Floating point value of non-centrality parameter.
8	Output address of $x_{(i)}$ and $F(x_i)$. SEQUENTIAL 1-15 and/or PRINT.

VI. References

[1] Darling, D.A., "The Kolmogorov-Smirnov, Cramer-Von Mises Tests," Annals of Mathematical Statistics, Vol. 28, (1957), pp. 823-838.

[2] Goodnight, James, Department of Experimental Statistics, N.C. State University, Raleigh, North Carolina. Mr. Goodnight developed and programmed the computational procedure used to integrate the central and non-central F distribution.

[3] Graybill, F.A., An Introduction to Linear Statistical Models, Vol. I, Chapter 4, "Distribution of Quadratic Forms" pp. 74-92.

[4] Lindgren, B.W., Statistical Theory, New York: MacMillan Company, (1962).

[5] Owen, D.B., Handbook of Statistical Tables. This book contains a table of the critical values for the Kolmogorov-Smirnov statistic. It is available for reference in the SOUPAC office.

FACTOR ANALYSIS PACKAGE

BINORMAMIN

I. General Description

BINORMAMIN ROTATION rotates a matrix, F, of orthogonal factor loadings to oblique simple structure.

It does this by iterating for T in FT = A (where A is the rotated factor pattern) so as to minimize:

$$K = \sum_p K_p = \sum_p \sum_{q=p}^{q} \left[\frac{\sum_j (v_{jp}^2/h_j^2)(v_{jq}^2/h_j^2)}{(\sum_j v_{jp}^2/h_j^2)(\sum_j v_{jq}^2/h_j^2)} \right]$$

Since solving directly for K is too complex, BINORMAMIN takes one vector at a time, rotating it against all the others, to minimize each K_p.

Its name comes from the fact that is uses a double (BI) NORMAlization in seeking a MINimum.

For further information see:

1. Kaiser, H. F. and Dickman, K. W., "Analytic Determination of Common Factors". Unpublished manuscript, University of Illinois, 1959.

2. Harmon, H. H., Modern Factor Analysis . Chicago, University of Chicago Press, 1960. pp.326ff.

II. Restrictions

Input is limited to matrices of 150 x 30 or less.

III. Parameters

After the program name, BINORMAMIN, are the following parameters:

Parameter Number	Use or Meaning
1	Input Address of factor matrix, F.
2	Output Address of factor matrix, F.
3	Output Address of the transformation matrix, T.
4	Output Address of the reference vector structure, V.
5	Output Address of the correlations between reference vectors.
6	Output Address of the primary factor pattern, P.

Parameter Number	Use or Meaning
7	Output Address of the correlations between factors.
8	Maximum number of iterations (see note). If blank, the maximum will be set at 100 iterations.
9	Convergence criterion (see note). A. Defined zero change: iterating will stop when each element in V changes by less than A. (A must be less than .2 and no less than .0000001). B. Defined zero rotation: iterating will stop when each vector in T changes by less than θ, where θ is the angle whose cosine is B. (B must be less than 1.0 and no less than .9). If left blank, A will be set to .001.
10	If an initial T is to be read in, input address of T.
11	Output Address of the initial T.

Note on Output: A. Any output option left blank will not be output.

B. The program will always print out the program name and the number of iterations actually done.

C. The program will print out the largest change in V, unless option 9B is used.

D. All data printed out is to 7 decimal places.

Note on parameters 8 and 9: Program will stop at whichever criterion it meets first.

Note on parameter 9: This parameter is a floating point constant and therefore must be enclosed in asterisks, with a decimal point, as in example:

Example: BINORMAMIN (CARDS)(('(SEQ 1/PRINT ('()(PRINT()*.0001*.

Store V on SEQ 1, also prints V and correlations between factors. On the last iteration, no element in V changed by more than .0001. unless the maximum of 100 iterations was reached.

CENTROID FACTOR ANALYSIS

I. General Description

CENTROID FACTOR ANALYSIS computes a set of f linearly independent vectors (factors) which are mutually uncorrelated. Normally, a factor analysis decomposes a matrix of correlations, R_n, into a set of f factors. The factors are arrayed as column vectors in the factor matrix, F, such that

$$R_n = FF' + R_{(n-f)}$$

where R_{n-f} is the matrix of residual effects. The K^{th} factor is computed by dividing the column sums of R_{n-k} by the square root of the total sum of elements of R_{n-k}

$$fi,k = \Sigma\, r_{i,j}^{(h)} \Big/ \sqrt{\Sigma\Sigma_{ij}\, r_{i,j}^{(k)}}$$

Between each factor extraction, the variables in the residual matrix are successively reflected until all the column sums are positive.

For more detailed discussion see:

1. L. L. Thurstone, Multiple Factor Analysis, Chicago, University of Chicago Press, 1947, pp. 149-175.

2. Harry Harmon, Modern Factor Analysis, Chicago, University of Chicago Press, 1960, pp. 192-215.

II. Restrictions

The input matrix for the CENTROID program must not exceed the dimensions of 190 x 190. The input matrix is further limited to being a square, positive definite or semi-definite, symmetric matrix. Commonly, correlation, covariance, or cross-product matrices are used as input data. Any attempt to introduce communality estimates (change the diagonal elements) must be made before data is passed to the CENTROID program. A set of communalities, which are incorrectly estimated, can make the matrix non-positive and could conceivably cause a hang-up.

The input data may come from any storage medium which conforms to SOUPAC. Similarly, the output codes follow the established conventions and are at the option of the user.

The input matrix may be completely factored (i.e., N factors from a N variable matrix). However, factoring may be stopped by any of three criteria:

1. The user may specify the number of factors to be extracted. This criterion provides an upper limit beyond which factoring will not be done. Consequently, it is advisable to put the maximum value on this limit in cases where it is not the primary criterion. (Set it equal to the number of variables).

2. The per cent of total variance removed from the R matrix is a second limiting criterion. This parameter also specifies an upper limit to the process. Therefore, it should be set at 100 per cent unless it is the criterion for stopping.

3. The last criterion is to stop when the factor contribution falls below 1. The use of this procedure is dictated by the presence or absence of its associated parameter.

If all three criteria are used simultaneoulsy, factoring will be stopped by whatever criterion is met first.

III. Parameters

Following the program anme on the program call card come the parameters needed by the program. The parameters must appear in the order below:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15.
2	Output Address. SEQUENTIAL 1-15 and/or PRINT.
3	Maximum number of factors to be extracted. This must be less than or equal to the order of the input matrix.
4	Per cent of total variance to be removed expressed as an integer between 0 and 100.
5	The presence of any number greater than 0 in this parameter indicates that factoring should stop when the factor contribution falls below unity.
6	Output Address of Residual Matrix.

If parameters 3 and 4 are left blank then by default option they will be set to maximum possible values and a message will be printed.

Residual Matrix must be stored before it can be printed.

Example: Assume that you have 77 variables and that the correlation matrix is stored on SEQ 1, then legal forms of CENTROID call statement may be:

```
CENTROID(SEQ 1)(PRINT)(77)(100)(1).
CENTROID  (SEQ 1)(P(F))(50)(80).
CENTROID (SEQ 1)(SEQ 2/P)(20)(100)(1)(SEQ 3/P).
CENTROID (SEQ 1)(SEQ 2/P(F)'(15)(90)()(SEQ 3/P(F)).
CENTROID (SEQ 1)(P).    In this case, number of factors = 77
and per cent of variance = 100 will be assumed by default.
```

COMMUNALITY ESTIMATION

I. General Description

Five methods of COMMUNALITY ESTIMATION are offered in this program. In each case the estimates replace the diagonal elements of the matrix. They are as follows:

Code Number	Method
1	The element of largest absolute magnitude in each row replaces the diagonal element of the row.
2	The square of the multiple R of each variable with all others replaces the diagonal entry for that variable. (See Special Comment Number 2).
3	Communalities produced from another analysis and are to be input from cards or another storage medium.
4	For each row (N) $\left(\left(\sum_{j=1}^{N} r_{i,j}^{2}\right)/N\right)^{1/2}$ replaces the diagonal entry for that row. This is the square root of the average square across the row.
5	For each row (N) $(r^{*}_{ik})(S_i - r^{*}_{ik})/(S_k - r^{*}_{ik})$ replaces the diagonal entry for that row where: $r^{*}_{ik} = \max \text{ abs } (r_{ij})$ and $S_i = \sum_j \text{abs } (r_{ij})$, $S_k = \sum_j \text{abs } (r_{kj})$ This method of COMMUNALITY ESTIMATION is due to Professor L. Tucker.

II. Restrictions

Input is restricted to correlation matrices of order 150 or less.

III. Parameters

The parameters for the COMMUNALITY ESTIMATION program appear on the program call card. They must follow the program name in this order:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15.

Parameter Number	Use or Meaning
2	Output Address. SEQUENTIAL 1-15 and/or PRINT.
3	Section Code Number. (See General Description).
4	Input Address if Option 3 is used.

IV. Special Comments

(1) If the correlation matrix and communalities both are input from cards, the correlation matrix precedes the communality estimations. (See Code Number 3 in the General Description).

(2) If the input correlation matrix is singular, or very nearly so, squared multiple correlations computed by standard procedures may be subje to considerable error, and will usually exceed unity for several variables For this reason, the user should be aware of the characteristics of the matrix. The program will check to see that all R^2 are less than or equal to 1.0. If this is not the case, execution will cease and a message will be printed stating that the correlation matrix is virtually singular. There is an alternative procedure devised by Ledyard Tucker for finding R^2 in singular matrices. Information is available in the SOUPAC office.

V. Reference

Harmon, Harry: Modern Factor Analysis, Chicago, University of Chicago Press, 1960. pp. 83-90.

ITERATIVE FACTOR ANALYSIS

I. General Description

A. Procedural

This routine, upon option, provides one of four iterative factorization methods:

1. Alpha factor analysis (AFA, Kaiser, 1962)
2. Canonical factor analysis (CFA, Rao, 1955, Harris, 1962)
3. Stepwise maximum likelihood factor analysis (MLFA, Lawley, 1940)
4. Iterative principal axis factor solution (IPRAX, Traditional)

All four methods have in common that communalities and factor loadings are estimated simultaneously. In three cases (AFA, CFA, IPRAX) the number of factors decision can be made beforehand by the user, or it can be left to the program, in which case appropriate modifications of Guttman's lower bound criterion will be used.

The four methods differ from each other in theory with respect to the defining criterion of optimization, and consequently they differ technically with respect to the matrix that is diagonalized in each case.

1. AFA (Kaiser)

Optimization criterion: maximize the alpha-reliabilities (Cronbach) of the retained factors. If the number of factors decision is left to the program, the Kaiser modification of the Guttman criterion will be used and all factors with positive alpha-reliability will be iterated upon.

The diagonalization is on the matrix C in

$$C = H^{-1} (R - U^2) H^{-1} \quad \text{so that } C = Q\theta Q'$$

where R is an nxn input matrix of covariances, H^2 is a diagonal matrix of communalities, $U^2 = I - H^2$ is a diagonal matrix of uniquenesses, Q is an nxm matrix of latent vectors corresponding to the m largest latent roots in θ which are used to recompute new estimates of H^2 through $F = H Q \theta^{1/2}$. An initial set of H^2 is provided by $I - (\text{diag}(R^{-1}))^{-1}$ which is equivalent to the squared multiple correlations of R if R itself is a correlation matrix.

Invariance under scaling: Kaiser has shown that the resulting factors will be invariant under scaling, i.e., if a covariance matrix R gives rise to a factor matrix F then the covariance matrix SRS will give rise to a factor matrix SF (S diagonal).

Behavior of latent roots, alpah reliabilities: the n-m rejected roots of C add to zero at each state (i.e., C is non-Gramian), the m accepted roots are simple functions of the alpha-reliabilities of the retained factors in F. These reliabilities will be output by this sub-program.

2. CFA (Rao, Harris)

Optimization criterion: maximize the correlations between m linear combination of the common parts of the variables with m linear factors that are canonically correlated (Hotelling) with the variables in the common factor space. If the number of factors decision is left to the program, the Harris modification of the Guttman criterion will be used, leading to a Gramian $R-U^2$ of minimum rank.

The diagonalization is on the matrix

$$C = U^{-1} (R - U^2) U^{-1} \quad \text{so that } C = Q\theta Q'$$

where $F = UQ\theta^{1/2}$ is used to recompute new estimates for U^2, retaining the m largest roots of C in θ. The notation is the same as in section 1 (AFA). An initial set of U^2 is provided by $[\text{diag } (R^{-1})]^{-1}$.

Invariance under scaling: the resulting factors are again invariant under scaling as defined in section 1 (AFA).

Behavior of latent roots: Chi-square criterion: Rao has shown that the n-m rejected roots approach unit at convergence. For exact rank m data they will be "exactly" unity within the tolerance of the convergence criterion ETA (see section III - B). For data containing random error their departure from unity provides a likelihood ratio test for the hypothesis that the population matrix $P-V^2 = GG'$, where P, V, G are population parameters corresponding to R, U, F in the sample, is rank m or less. A criterion for this test is computed by this sub-program which can be compared with two chi-square approximations which are also output by this sub-program. Note that such a chi-square test is valid only if the iterative process has indeed converged, as indicated by the maximal discrepancy between trial vectors which is printed out for that purpose.

3. MLFA (Lawley)

The CFA variant of the program can be used for a step-wise maximum likelihood factorization in the Lawley-Rao sense.

Optimization criterion: maximize the likelihood function corresponding to the multivariate normal distribution with covariance matrix parameters $P = GG' + V^2$ (as defined in section 2, CFA), given the sample matrix R, under choice of G and V^2 and observing the side-conditions that $P-V^2$ is Gramian and V^2 diagonal with $0 < v_i < 1$.

The diagonalization is the same as in CFA, section 2, hence, the resulting factors are again invariant under scaling as defined in section 1 (AFA).

In contrast to CFA, however, the number of factors decision is made on statistical ground. The user would start with a reasonable guess for m (preferably $m < n/2$ to ensure positive degrees of freedom for the chi-square test, which otherwise will be bypassed). After convergence has been obtained, the user would insepct the chi-squared statistic. If the statistic is below the table values of the chosen porbability level (.05 or .01), then the hypothesis can be accepted at this level with corresponding risk and the user has the option to reduce m for a second run, etc. On the other hand, if the adjusted statistic exceeds the table value, then m must be raised until the adjusted statistic warrants acceptance of the hypotheses.

Within the package the user is free to re-enter the routine repeatedly with sequentially de- or increasing m specified on the call card. Since the test assumes convergence, it is pertinent that the number of iterations be allowed large enough for convergence to occur within the chosen tolerance bound ETA (see section III - B).

4. IPRAX (traditional)

Optimization criterion: none

The diagonalization is on the matrix

$$C = R - U^2 \quad \text{so that} \quad C = Q\theta Q'$$

where $F = Q\theta^{1/2}$ is used to recompute $H^2 = I - U^2$, retaining the m largest roots in θ. The notation is the same as in section 1 (AFA). An initial set of H^2 is provided by the identity matrix. If the number of factors decision is left to the program, the unmodified Guttman criterion will be used, i.e., all factors corresponding to roots of the input matrix R which exceed unity will be retained.

Invariance under scaling: as defined in section 1 (AFA) is not obtained by this method.

The behavior of the latent roots is not known at present. No statistical or other significance can be attached to the m largest or n-m smallest root of C.

5. Both covariance matrices and correlations matrices are acceptable as input. If covariances are used the tenth parameter should be 1. In this case the covariance matrix is scaled into a correlation matrix, and all computations, in particular the number of factors decision, are based on this correlation matrix. At the final stage the factors are scaled back so as to account for the covariance matrix which was input. The matrix of residuals is computed in the metric of the covariances.

References:

Guttman, L. "General Theory and Methods for Matrix Factoring," Psychometrika, 1955, IX, 1-16.

Harris, C. L. "Some Rao-Guttman Relationships," Psychometrika, 19[illegible], XXVII, 247-263.

Hotelling, H. "The Relation of The More Multivariate Statistical Methods to Factor Analysis," British Journal of Statistical Psychology. 1952, X, 69-79.

Kaiser, H. F. and Caffres, T. "Alpha Factor Analysis", Psychometrika, Vol. 30, 1965, p. 1-14.

Lawley, D. N. "The Estimation of Factor Loadings by the Method of Maximum Likelihood," Procedures of the Royal Society of Edin., 1940, LX, 64-82.

Rao, C. R. "Estimation and Tests of Significance in Factor Analysis," Psychometrika, 1955. XX, 93,111.

II. Restrictions

Input is restricted to matrices of order 100 x 100 or less. Up to 50 factors can be handled by this program. If the number of factors decision is left to the program and more than 50 factors are estimated, an appropriate message will be printed out and control will be returned to the system.

III. Parameters

The program name is ITERATIVE FACTOR ANALYSIS. After the name on the call card the parameters must appear in the following order:

Parameter Number	Use or Meaning
1	Input Address of correlation or covariance matrix. CARDS or SEQUENTIAL 1-15.
2	Output Address of correlation or covariance matrix. SEQUENTIAL 1-15 and/or PRINT.
3	Output Address for principal axis factors. SEQUENTIAL 1-15 and/or PRINT.
4	Output Address of residual matrix. SEQUENTIAL 1-[illegible] and/or PRINT.
5	Option code 0 if IPRAX 1 if ALPHA 2 if CANONICAL 3 if STEP-WISE MAXIMUM LIKELIHOOD

Parameter Number	Use or Meaning
6	Maximum number of cycles to be executed. If left blank, 50 cycles will be used as upper limit.
7	Number of factors to be extracted. If left blank, all factors with roots exceeding unity will be retained.
8	Exponent of convergence, n, where tolerance ETA = 10^{-n}. If left blank n = 3 or ETA = 10^{-3}. If all goes well, the program will stop as soon as either one of the stopping criteria is met. Error stops, if they occur, are labelled accordingly.
9	Sample size (for CFA only). If left blank, the chi-square computations are by-passed. If specified, chi-square is computed with the sample size.
10	1 if input matrix was a covariance matrix.

A. Output common to all four sub-programs

1. Matrix output within system conventions:

   a. R (input covariance matrix)
   b. F (factor matrix)
   c. R-FF' (residual matrix)

2. Vector output, print only:

   a. communality vector (last iteration)
   b. vector of latent roots of C (last iteration)

3. Constant, print only:

   a. number of iterations completed
   b. largest discrepancy between trial vectors (H^2, U^{-1}, H^{-1}, depending on sub-program)
   c. root mean square of off-diagonal residual matrix
   d. per cent of variance removed

B. Additional output specific to sub-programs

AFA: The alpha-reliabilities of the m retained factors

CFA: chi-square statistic, chi-square approximations (Wilson, Hilferty) for p = .05 and p = .01, for comparison with statistic. Degrees of freedom.

IV. Special Comments

The accuracy should be approximately 6 digits in computations, possibly somewhat lower for a very large number of iterations. The effective accuracy depends on the chosen tolerance ETA and the actual convergence as indicated by the largest discrepancy between trial vectors. The chi-square approximations are within 2×10^{-2} for more than 8 degrees of freedom.

JACOBI

I. General Description

This program calculates eigenvalues and eigenvectors of a square, symmetric matrix, using the JACOBI rotating technique. This program is limited to matrices of 110 rows and 110 columns. The user should realize that this technique is extremely slow on large matrices, while the program takes no longer than PRINCIPAL AXIS FACTOR ANALYSIS for small matrices (up to 20 x 20).

II. Parameters

Parameter Number	Use or Meaning
1	Input Address of correlation matrix. CARDS or SEQUENTIAL 1-15.
2	Output Address of eigenvectors.
3	Output Address of principal axis factor.
4	Output Address of eigenvalues, stored as a row vector. PRINT is not valid. The eigenvalues are always printed.
5	Number of eigenvectors (or factors) to be output.

III. Special Comments

The eigenvalues are stored in descending algebraic order (from largest to smallest), and the eigenvectors and factors are placed in the same order.

IV. Reference

Ralston, A. and Wilf, H. S.: Mathematical Methods for Digital Computers, John Wiley and Sons, New York, 1964.

OBLIMAX ROTATION

I. General Description

The OBLIMAX OBLIQUE ROTATION transforms a set of factors F to a new set V such that the factor kurtosis,

$$K = \frac{\Sigma\Sigma\, v_{ij}^{4}}{(\Sigma\Sigma\, v_{ij}^{2})^{2}} \qquad \begin{matrix} i = 1, 2, \ldots\ldots, n \\ j = 1, 2, \ldots\ldots, k \end{matrix}$$

is at a maximum.

The purpose of the transformation is to attempt to rotate analytically to a position similar to that obtained by applying Thurstone's rules for simple structure. (See Multiple Factor Analysis, L. L. Thurstone, 1947, pp. 319-410.) However, Thurstone's rules and the oblimax procedure are not the same, and it is too much to expect that results obtained from both procedures will agree exactly.

It would be desirable to solve directly for the transformation matrix T, but unfortunately no solution to this problem has been found. Instead oblimax takes two vectors at a time, solves for the rotational angles, transforms the vectors, and then selects another pair until all $k(k-1)$ pairs have been rotated. This process is repeated iteratively until the criterion K no longer increases. Despite the pairwise procedure, K is well behaved, and in general, approaches steadily to a minimum.

For any pair of factors, a and b, the solution proceeds as follows:

$$K = \frac{\Sigma\Sigma(a_i \cos \phi_j + b_i \sin \phi_j)^4}{[\Sigma\Sigma(a_i \cos \phi_j + b_i \sin \phi_j)^2]^2} = \frac{\Sigma\Sigma(a_i + b_i X_j)^4}{[\Sigma\Sigma(a_i + b_i X_j)^2]^2}$$

The derivative of K_{ab} is set equal to zero, resulting in a quartic equation in X which is tan ϕ. Two solutions for X will be maxima, and, each X is found, the sign of the second derivation is inspected to select maxima. A small transform (2 x 2) is created, but before post-multiplication is performed, the transforms must be adjusted so that when it becomes a part of T, t_a, and t_b, will remain normalized. In this way, both B and T are developed pair by pair.

For references see:

Pinzka, C., and Saunders, D. R., "Analytic Rotation to Simple Structure II: Extension to an Oblique Solution." Research Bulletin RB-54-31. Princeton, N. J.: Educational Testing Service, 1954.

II. Alternate Use

If the user already has a transformation matrix, he may use it to compute V_{rs} et.al by giving the input address of T in parameter 8; in this case,

the oblimax calculation of T will be skipped. If both F and T are to be input from cards, then the data deck of F should precede the deck of T.

III. Output

The OBLIMAX program always prints the following (unless parameter 8 is used):

1. The value of K for each pass
2. The iteration time

It outputs the following on demand (See Parameters):

1. Transformation matrix T
2. Reference vector structure, $V_{rs} = FT$
3. Reference vector correlations, $C_{rs} = T'T$
4. Diagonal of D and of D^{-1}
   where D is the diagonal matrix of the reciprocal square root of the diagonal elements of C_{rs}^{-1}
5. Primary factor pattern, $V_{fp} = FTD^{-1} = V_{rs}D^{-1}$
6. Primary factor correlations, $C_{fp} = DC_{rs}^{-1}D$

All data is printed out to seven decimal places.

IV. Restrictions

The number of variables plus the number of factors must be no more than 300.

V. Note on Parameter 2

If row normalization is specified, the normalization constants will be preserved and the rows will be rescaled to proper length after rotation and prior to output.

VI. Parameters

The program name, OBLIMAX, appears first on the program call card and is followed by the following parameters. Any output option (except parameter 9) may be SEQUENTIAL 1-15 and/or PRINT: it may be left blank if not desired.

Parameter Number	Use or Meaning
1	Input Address of F. CARDS or SEQUENTIAL 1-15.

Parameter Number	Use or Meaning
2	If rows are to be normalized before rotation, punch a 1; otherwise a zero or leave blank. (See Note on Parameter 2).
3	Output Address of T.
4	Output Address of V_{rs}
5	Output Address of C_{rs}
6	Output Address of V_{fp}
7	Output Address of C_{fp}
8	Input Address of T (See Alternate Use).
9	D-value and Inverse of D. PRINT only.

ORTHOGONAL PROCRUSTES; KAISER'S TECHNIQUE FOR RELATING FACTORS BASED ON DIFFERENT SAMPLES

I. General Description

This program offers 2 options.

1. Orthogonal Procrustes. Given 2 matrices, a factor matrix A and a target matrix B, the program solves

$$AT = B + E$$

for T in a least squares sense (i.e., minimizing tr(E'E), under the restriction that the transformation matrix T by orthonormal, i.e.

$$T'T = TT' = I.$$

There are no restrictions on A and B other than conformability. In particular, the method does NOT require full column rank in either of these input matrices.

For further information and some applications, see

> Schönemann, P. H., "A Solution of the Orthogonal Procrustes Problem with Applications to Orthogonal and Oblique Rotation," Unpublished Ph.D. thesis, 1964, (on file).

2. Kaiser's Technique for Relating Factors Based on Different Samples. Given 2 factor studies based on different samples but overlapping in some (not necessarily all) variables, this technique yields, under certain mild (non-singularity) conditions and certain strong assumptions, a matrix of estimated cosines between the two sets of factors which might be interpreted as correlations in a very loose sense ("quasi correlations"). Operationally

$$R_{12} = R_{1c} R_{cc'} R_{c'2}$$

where R_{c1} and $R_{c'2}$ give the correlations of orthogonal reference axes (such as centroid or principal axis factors) with primary factors in each study, $R_{cc'}$ is the transformation matrix obtained when fitting one set of tests to the other by the Orthogonal Procrustes technique above, and R_{12} is the matrix of quasi correlations relating the factors of one study to those in the other. This matrix need not be square. The program, as written, assumes centroid (or, equivalently, principal axis) matrices and reference vector structures as input and proceeds to compute R_{c1} and $R_{c'2}$ from those, assuming this to be most convenient for most users. Note that the essential information is embodied in the matrix $R_{cc'}$, which is identical with the matrix T of the orthogonal Procrustes routine, so that the remaining algebra can also be performed by other programs in SOUPAC, if more convenient.

For further information, see:

Kaiser, H. F. "Relating Factors Between Studies Based Upon Different Individuals." Unpublished MS., U. of I., 1960.

II. Restrictions

Input is restricted to matrices of order 100 x 50 or less.

III. Parameters

The program mnemonic, ORT, appears first on the card and is followed by up to 13 parameters

Parameter Number	Description	Orthogonal Procrustes	Kaiser's Factor Matching
1	Input Address CARDS or S1-S15	A	$_1R_{tc}$
2	Input Address CARDS or S1-S15	B	$_2R_{tc}$
3	Output Address S1-S15 and/or PRINT	A	$_1R_{tc}$
4	Output Address S1-S15 and/or PRINT	B	$_2R_{tc}$
5	Output address S1-S15 and/or PRINT	T	
6	Output address S1-S15 and/or PRINT	AT	
7	Output Address S1-S15 and/or PRINT	E	
8	Input Address CARDS or S1-S15		$_1R_{tn}$ (reference vector structure for study 1)
9	Input address CARDS or S1-S15		$_2R_{tn}$ (reference vector structure for study 2)
10	Output Address S1-S15 and/or PRINT; Blank if not desired		$_1R_{tn}$
11	Output address S1-S15 and/or PRINT; Blank if not desired		$_2R_{tn}$

Parameter Number	Description	Orthogonal Procrustes	Kaiser's Factor Matching
12	Output address S1-S15 and/or PRINT.		R_{12} ("quasi correlations")
13	Input address of integer k.		needed only if there are m factors in study 1 and k < m factors in study 2.

Note: If parameter 8 is empty, an Orthogonal Procrustes Solution will be computed and control will then be returned to the system. If an input address for ${}_1R_{tn}$, the reference vector structure of the first study, is given, it will function as a switch and invoke the subroutine for Kaiser's technique. In this case, the reference vector structures should be in the same order as the centroids. All 4 matrices should contain only variables which were common for both studies.

Rectangular R_{12}: If the number of factors were different in both studies, say m > k, then the matrices with m factors should each precede the corresponding matrix with k factors. For brevity, let this input sequence be denoted by F_m, F_k, V_m, V_k. F_m, V_m, and F_k should be read in with a format for m columns, e.g. its data card might read "DATA(m) (mF5.3)" so that its machine image will be augmented by m-k columns of zeros (so as to allow for an Orthogonal Procrustes fit of the centroids). On the other hand, V_k is to be read with a format signaling k columns, e.g. its data card might read "DATA(k) (kF5.3)" (so as to allow for inversion of $V_k'V_k$). Finally, the 13th parameter should be k in this case.

SCATTER PLOTS

I. General Description

This program generates 2 different types of one-page plots depending on the type of data input. If only one row of data is input, the program generates a successive value plot. That is, a single row of data will be interpreted as representing successive values of a single variable and will be plotted on the vertical axis, while the column number (value number) is plotted on the horizontal axis. For example, this feature is especially suitable for plotting a row of eigenvalues output from a factor analysis program.

If more than one row of data is input, the program generates bivariate scatter-plots , assuming that columns represent variables. The program will generate a separate one-page scatter-plot for each possible pair of variables, up to a limit specified by the user. The user cannot specify particular pairs to be plotted. The first variable of each pair is plotted on the horizontal axis and the second is plotted on the vertical axis.

II. Options

There are four options available in this program.

1) Bounds on axes: The user may specify upper and lower bounds for either or both axes. If the user does not specify these limits the program will generate its own bounds for the axes. Any point lying beyond user-specified bounds will be omitted from the plot.

2) Printed Axes: The user may specify whether he wants printed axes included in a bivariate scatter-plot . Since these axes must be printed at the center of the graph, they should probably be omitted if the origin of the plot to be generated will not also be centered on the graph.

3) Point Identification: The user may specify whether points in the plot should be individually identified or plotted as x's. There are 60 identification characters available; if the user requests identified points, the first 60 points will be identified as follows:

POINT IDENTIFICATION TABLE

Point	ID	Point	ID	Point	ID	Point	ID
1	A	16	P	31	5	46	K
2	B	17	Q	32	6	47	L
3	C	18	R	33	7	48	M
4	D	19	S	34	8	49	N
5	E	20	T	35	9	50	O
6	F	21	U	36	A	51	P
7	G	22	V	37	B	52	Q
8	H	23	W	38	C	53	R
9	I	24	X	39	D	54	S
10	J	25	Y	40	E	55	T
11	K	26	Z	41	F	56	U
12	L	27	1	42	G	57	V
13	M	28	2	43	H	58	W
14	N	29	3	44	I	59	X
15	O	30	4	45	J	60	Y

Obviously, if there are more than 35 points on a given plot, there will be some duplicate identification characters. It remains for the user to differentiate these points by virtue of their location on the plot.

If the identification is requested, the program will plot points 61-85 as asterisks, but points beyond #85 will be omitted entirely. In general, if a user wishes to plot more than 60 points, it is not recommended that he specify identification of points.

If the user specifies counted points, the program will plot an x at the location of each point. If there are 2 or more points at a specific location, the program will plot a number (2, 3, ..., 9) indicating how many points are present at the location. If more than 9 points exist at a specific location, the program will plot the letter M at that location.

4) Variables to be plotted: The user may specify the number of variables for which he wants pairwise scatter plots. The program will generate bivariate plots for all possible pairs of the set of variables specified. For example, if the user inputs a matrix of 10 eigenvectors and sets this parameter (#2) equal to 4, the program will generate plots for eigenvectors 1 vs 2, 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4, and 3 vs 4.

III. Input

Input must be either (1) a single row of data to be converted to a successive value plot, or (2) a data-matrix to be converted to bivariate scatter-plots.

IV. Special Comment

Due to the various problems involved in plotting points on standard computer output paper, these scatter-plots will be of limited accuracy. The most obvious problem is that there are a finite number of locations at which a given character can be printed, resulting in slight misplacement of many points. Thus, though the plots will present a valid and useful picture of the general relationships present in the data, the user should be cautious in any technical or detailed analysis of these graphs.

V. Parameters

The program mnemonic (PLO) appears first on the program card and is followed by up to 7 parameters; only the first parameter is required. Parameter 1 is an address; parameters 2 and 3 are integers enclosed in parentheses; parameters 4-7 are floating point numbers enclosed in asterisks. Parameters 4 and 5 should be either both specified or both blank. Also, parameters 6 and 7 should be either both specified or both blank.

Parameter Number	Use or Meaning
1	Input address (cards or S1-S5)
2	Number of variables to be plotted. (The value of this parameter is irrelevant if input is a single row. If this parameter is omitted in a bivariate scatter-plot program, all possible pairs of variables will be plotted.)
3	Code for point identification and inclusion of axes; (-1) axes omitted; points counted (-2) axes omitted; points identified (1) axes included; points counted (2) axes included; points identified Default is (-1).
4	Upper bound of X-axis
5	Lower bound of X-axis
6	Upper bound of Y-axis
7	Lower bound of Y-axis

VI. Example

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
CORRELATIONS (CARDS) (P) (S1/P).
```

VI. Example (continued)

```
PRINCIPAL AXIS (S1) (S2/P) (10) (100) (0) (S3/P) (S4).
PLOT (S2) (4) (2).
PLOT (S3) (4) (2) *1* *-1* *1* *-1*.
PLOT (S4).
ENDS
DATA (10) (10F3.0)
 .
 .  data deck
 .
END#
/*
```

This program would do the following: compute a correlation matrix for the data on cards; factor the correlation matrix, storing 10 factors on S2, 10 eigenvectors on S3, and a row of eigenvalues on S4; PLOT all pairs of the first 4 factors, with axes printed, points identified, and the program setting its own bounds for the axes; PLOT all pairs of the first 4 eigenvectors with axes printed, points identified, and bounds of +1 and -1 for both axes; PLOT the row of eigenvalues as a successive value plot with axes omitted and points marked as X's.

*This program was developed from a program written by F. W. Young of the University of North Carolina.

PRINCIPAL AXIS FACTOR ANALYSIS
(Eigenvalues and Vectors)

I. General Description

The purpose of PRINCIPAL AXIS FACTOR ANALYSIS is to determine a factor matrix, F, given a Gramian matrix, R, of order n such taht

$$F_{(n,f)}F'_{(f,n)} = R^*_{(n,n)}$$

where R* is an approximation to R.

The column vectors of F are defined as the factors (measures of dimensionality) of the original matrix, R. The solution for the matrix F is the classical eigen problem. Consequently, the computations are done by an eigenvalue subroutine. Before output the eigenvectors, E_j, are scaled as follows:

$$F(I,J) = E(I,J)*LAMBDA(J)**.5$$

for I = 1,....,n. J = 1,....,n.
to generate the principal axis factors, F. (See Introduction on Factor Analysis).

For a more detailed discussion see:

Harry Harmon, Modern Factor Analysis, Chicago, University of Chicago Press, 1960, pp. 154-191.

II. Restrictions

The input matrix for the PRINCIPAL AXIS program must not exceed the dimensions of 190 x 190 double precision. The input matrix is further limited to being a square, symmetric matrix. Generally correlation, covariance, or cross-product matrices are used as input data. It should be noted that matrices with large numerical entries such as cross-products may generate output values which cannot be printed under the fixed output formats. The probability of this happening is very small. Any communality estimation (i.e., change in the diagonal entries of R) must be done prior to the input of R, to the PRINCIPAL AXIS program.

If the communality estimates are used, the user should check the resulting roots for negative numbers. If any exist the associated vector is meaningless.

The input data may come from any source conforming to SOUPAC. Similarly, the output codes follow the established conventions and are specified at the option of the user.

The R matrix may be completely factored (i.e., N factors from N variable matrix). However, there are three criteria which may be used to stop the factoring:

1. The user may specify the number of factors to be extracted. This criterion provides an upper limit beyond which factoring will not proceed. Therefore, it is necessary to put the maximum value in this limit in cases where it is not the primary criterion.

2. The percentage of total variance removed from R is the second limiting criterion. This parameter also specifies an upper limit to the process. Therefore, it should be set at 100 per cent unless it is the criterion for stopping.

3. The last criterion is to stop when the factor contribution (eigenvalue or root) falls below 1. The use of this procedure is dictated by the presence of its parameter.

If all three criteria are employed simultaneously, factoring is stopped by whichever criterion is first met.

III. Parameters

The parameters for the PRINCIPAL AXIS program appear on the program call card. They must follow the program name in this order:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15.
2	Output Address. SEQUENTIAL 1-15 and/or PRINT.
3	Maximum number of factors to be extracted. This must be less than or equal to the order of the input matrix.
4	The percentage of total variance to be removed expressed as an integer between 0 and 100.
5	The presence of a number greater than 0 indicates the factoring should stop when the eigenvalues (roots) fall below unity.
6	Output Address of Eigenvectors
7	The address of where eigenvalues are to be placed as a row vector if they must be stored for further use. If values need not be saved, leave parameter blank. PRINT is not valid.

Parameter Number	Use or Meaning
8	Mode of sorting eigenvalues and associated vectors. The codes are as follows:

Code	Meaning
0	Descending algebraic order
1	Descending absolute values
2	Order of extraction
10	Ascending algebraic order (the k smallest root)
11	Ascending absolute values
12	Reverse order of extraction

Leaving any parameter blank is the same as specifying zero. Consequently, options which are not needed can be avoided by leaving the associated parameter blank.

PROCRUSTES (Oblique Case)

I. General Description

This program offers 3 options:

1. (Oblique) Procrustes. Given A, B, the program solves

$$AT^* = B + E$$

for T^* in a least square sense (i.e., minimizing tr[E'E]), so that

$$T^* = (A'A)^{-1}A'B,$$

and then normalized T^* by columns to yield $T = T^*D$ so that diag $(T'T) = I$. It then computes AT which, in a loose sense, can be regarded as a least squares fit to A to B under the restriction that diag $(T'T) = I$. It also provided $C_f = D_n(T'T)^{-1}$ where D_n is a normalized diagonal matrix so that diag $(C_f) = I$. If D gave the cosines between tests C_f will give the factor intercorrelations. A has to be a full column rank.

2. Dwyer Extension Analysis. Given $F = R_{tc}$, a centroid or equivalent matrix of cosines between tests t and uncorrelated factors c, and $L = R_{cn}$, a matrix of cosines between uncorrelated factors c and uncorrelated reference vectors n, this program computes

$$Q = T_{tn} = F(F'F)^{-1}L$$

which is used as a post-multiplier on some correlations matrix R_{et} between the tests t x in F and some set of extension variables e given R_{en}, the cosines of the extension variables e with reference n, to the extent that the former can be projected into the sub-space spanned by the latter. This multiplication

$$R_{en} = R_{et}\ T_{tn}$$

can be performed by use of the MATRIX program.

3. Left Inverse (transposed). Given A, the program will return

$$Q = A(A'A)^{-1}$$

provided A was a full column rank. Q is the transposed left inverse of A which can be used in lease squares application.

II. Restrictions

Input is restricted to matrices (A, B, or F) of order 190 x 50 or less.

III. Parameters

The parameters for this program appear on the program call card. They must follow the program name in this order:

Parameter Number	Use or Meaning	Procrustes	DEA	LINV
1	Input Address CARDS or SEQUENTIAL 1-15.	A	F	A
2	Input Address CARDS or SEQUENTIAL 1-15.	B	L	
3	Output Address SEQUENTIAL 1-15 and/or PRINT.	A	F	A
4	Output Address SEQUENTIAL 1-15 and/or PRINT.	B	L	
5	Output Address SEQUENTIAL 1-15 and/or PRINT.	T	Q	$A(A'A)^{-1}$
6	Output Address SEQUENTIAL 1-15 and/or PRINT.	C_f		
7	Output Address SEQUENTIAL 1-15 and/or PRINT.	AT		
8	Output Address SEQUENTIAL 1-15 and/or PRINT.	E		
9	Choice Address	0	1	2

SQUARE ROOT FACTOR ANALYSIS

I. General Description

The SQUARE ROOT method of factor analysis, also called the Diagonal Method, by L. L. Thurstone, decomposes a correlation matrix R (or any other positive semi-definite or definite symmetric matrix) such that

$$R = F\,F' + R(k+1)$$

where R(k+1) is the residual matrix after extracting k factors. Of course if all n factors are extracted, the residual matrix becomes a null matrix.

The factor f_j is computed by dividing each element of the j^{th} column of R by its diagonal square root:

$$f_{ij} = r_{ij}/\sqrt{r_{jj}} \quad (i = 1,2,.....,n)$$

The matrix $A = f_j \cdot f'$ is then subtracted from R and the operation repeated on the residual matrix.

Prior to the widespread use of high speed computers, the SQUARE ROOT method was sometimes used as a substitute for the PRINCIPAL AXIS or CENTROID method due to the relative ease of computing a square root factor. When used in this way, one seeks to extract the maximum variance for each factor, in which case Parameter 4 should be blank. The program then selects the next column on the basis of the largest residual column sum of squares.

Nowadays, however, the SQUARE ROOT method is more likely to be used for special purposes. By selecting successive pivot variables, the user retains control over the factoring. Factors are passed directly through the test variables and the effect of these variables is removed from the matrix. The communalities or row sums of squares are the squared multiple correlations of the remaining variables with the pivot variables.

The pivots selected may be any columns in the matrix. Let us assume, however, that these are adjacent to each other in the upper right hand corner of the partitioned matrix below:

$$R = \begin{bmatrix} R_{pp} & R_{ps} \\ R_{sp} & R_{ss} \end{bmatrix}$$

Then the effect of pivoting successively on the variables in the upper right hand corner is shown by the residual matrix as follows:

$$R^{(p+1)} = R - F_r F_p' = \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & R_{ss} - R_{sp} R_{pp}^{-1} R_{ps} \end{array}\right)$$

II. Restrictions

A. Dimension

Maximum size of the R matrix is 190 variables.

B. Special Conditions

1. The researcher may specify the extraction of any number of factors up to dimension of R.

2. The researcher may specify the diagonal element to be used in the extraction of each factor, or he may have the procedure remove the maximum variance each time.

3. The residual matrix may be saved if the researcher desires.

III. Parameters

Following the program name the parameters must appear in the following order on the program call card:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15. (Correlation or positive definite or semi-definite matrix).
2	Output Address. SEQUENTIAL 1-15 and/or PRINT.
3	Number of factors extracted.
4	Input Address for row vector specifying order of diagonal elements to be used in factor extraction. Optional. CARDS or SEQUENTIAL 1-15.
5	Output Address for residual matrix. SEQUENTIAL 1-15 and/or PRINT.

IV. Special Comments

If the diagonal element for each factor is specified, and if both input addresses are cards, then data precedes diagonal specification.

V. Example

Assume you have a 20 x 20 correlation matrix on cards and that you want to extract 15 factors; also you are reading the pivot column from cards. The program would be set up as follows:

```
/*ID
//  EXEC  SOUPAC
//SYSIN  DD  *
SQU(C)(P)(15)(C)(P).
END SOUPAC
DATA(20)(8F9.7)
     .
     .    data
     .
END #
DATA(15)(15I2)
..................................... diagonal specification card(s)
END #
/*
```

THREE-MODE FACTOR ANALYSIS

I. General Description

A. GENERAL COMMENTS

This program provides a factor analytic solution for a 3-dimensional i by j by k data matrix. The computational procedures employed are those presented in Method III of Tucker's article (reference below). This method provides most efficient analysis when one of the modes, usually individuals is quite large, though this is certainly not a necessary condition.

B. THE THEORETICAL MODEL

Here, i, j, and k represent the modes of classification which are directly related to the observation of the data; i, j, and k are thus termed observational modes. An example would be the observation of scores for i individuals on j tests given under k different conditions.

Through factoring, we wish to reduce the observational modes i, j, and k to corresponding derivational modes m, p, and q. Each of the derivational modes can be thought of as a set of factors in the domain of the corresponding observational mode. The core matrix G then serves to describe the relationships among the derivational modes.

The fundamental three-mode factor analysis model is represented by the equation:

$$\tilde{x}_{ijk} = \sum_m \sum_q \sum_p a_{im}\, b_{jp}\, c_{kq}\, g_{mpq} \quad ,$$

where $\tilde{x}_{ijk}$ is an approximation to the observed score x_{ijk}; a_{im}, b_{jp}, and c_{kq} are entries in two-mode matrices ${}_iA_m$, ${}_jB_p$, and ${}_kC_q$ describing the elements in the observational modes i, j, and k in terms of the dimensions in the derivational modes m, p, and q respectively; the coefficients g_{mpq} are entries in a three-dimensional matrix G and represent the measures of the phenomenon being observed for each combination of the dimensions of the derivational modes.

In matrix form, the model could be represented as:

$${}_iX_{jk} = {}_iA_m\, G_{(pq)} ({}_pB_j \boxtimes {}_qC_k) \quad ,$$

where $\boxtimes$ indicates a Kronecker product. Matrices A, B, and C are factor solutions for modes i, j, and k respectively which serve to transform the core matrix G of the 3 derivational modes to the matrix X representing the 3 observational modes.

C. INPUT DATA

The input data must be a Gramian matrix, usually correlations, covariances, or cross-products, in the form $_{jk}R_{jk}$, where i is assumed to be the largest mode and jk represents the combination mode, with mode k nested within mode j.

II. Output

The output consists of the following:

(1) the $_{j}P_{j}$ and $_{k}Q_{k}$ matrices which represent the correlations, covariances or cross-products within modes j and k respectively;

(2) the eigenvalues and principal axis factors of $_{jk}R_{jk}$;

(3) the eigenvalues and eigenvectors of $_{j}P_{j}$;

(4) the eigenvalues and eigenvectors of $_{k}Q_{k}$;

(5) the core matrix $_{pq}G_{m}$, where m, p, and q represent the derivational modes corresponding to observational modes i, j, and k respectively.

All of this is printed out and may also be stored on sequential storage devices. The user must specify the number of factors to be extracted from each of the three modes. This procedure is employed since the use of other factor-stopping criteria (e.g. per-cent of variance accounted for, or eigenvalues below unity) could easily lead to the computation of a great many useless factors as well as a very large and unmanageable core matrix. The user is also cautioned against specifying large numbers of factors since this would cause substantial increases in time required to factor the various modes and compute the core matrix.

It is strongly recommended that the user be familiar with the Tucker article and with factor analysis in general before attempting to use this program.

III. Parameters

The program mnemonic (T-M) appears first on the program card and is followed by the following 17 parameters, the first 8 of which are required. Output addresses are optional. All output is printed.

Parameter Number	Description
1	Input address of R matrix in the form $_{jk}R_{jk}$ (Cards or S1-S15).
2	Number of subjects or elements in mode i.

Parameter Number	Description
3	Number of variables in mode j.
4	Number of variables in mode k.
5	Number of factors to be removed from matrix R (mode i).
6	Number of factors to be removed from matrix P (mode j).
7	Number of factors to be removed from matrix Q (mode k).
8	Scratch address (see special comment).
9	Output address for row vector of eigenvalues of R.
10	Output address for principal axis factors of R.
11	Output address for matrix ${}_{j}P_{j}$.
12	Output address for row vector of eigenvalues of P.
13	Output address for eigenvectors of P.
14	Output address for matrix ${}_{k}Q_{k}$.
15	Output address for row vector of eigenvalues of Q.
16	Output address for eigenvectors of Q.
17	Output address for core matrix ${}_{pq}G_{m}$.

IV. Special Comment

The three-mode factor analysis program requires three separate interval storage areas. SOUPAC, however, has only two such areas available within its programs. Thus, the user must supply a scratch address. This can be any sequential file (S1-S15) not used as another parameter in the three-mode program. Any data previously stored on this file will be destroyed.

V. Reference

Tucker, Ledyard R. Some mathematical notes on three-mode factor analysis. Psychometrika, 1966, 31, 279-311.

UNRESTRICTED MAXIMUM LIKELIHOOD FACTOR ANALYSIS

Parameter Number	Use or Meaning
1	Input Address for correlation matrix. SEQUENTIAL 1-15; CARDS are not permitted
2	Output Address for final unrotated factor matrix. SEQUENTIAL 1-15. See also Parameter 13.
3	Input Address for row vector of initial estimate of uniqueness. CARDS, SEQUENTIAL 1-15 (optional).
4	Lower bound for number of factors.
5	Upper bound for number of factors.
6	Sample size (number of observations) on which correlation matrix is based.
7	Maximum number of iterations.
8	Probability of chance occurance, i.e., *1.00*.
9	1 to print input correlation matrix and partial correlation matrices after any variables have been removed.
10	1 to print technical output.
11	1 to print intermediate results.
12	1 to punch unrotated factor matrices.
13	1 to apply a varimax rotation to all factor matrices. If this parameter is used the output of parameter 2 will be a rotated factor matrix.

This program has been taken directly from Jöreskog (1967) with his permission. Anyone interested in the methods is referred to the references listed below. The program is temporarily limited to 75 variables and 30 factors. Parameters 1, 4, 5, 6, 7, and 8 are required. Parameter 8 must be enclosed within asterisks, **, and must have a punched decimal point.

References:

Jöreskog, K. G. UMLFA - a computer program for unrestricted maximum likelihood factor analysis. Research Memorandum 66-20. Princeton, New Jersey: Educational Testing Service. Revised Edition, 1967.

Jöreskog, K. G. Some contributions to maximum likelihood factor analysis. Psychometrika, 1967, 32, 443-482.

VARIMAX FACTOR ROTATION

I. General Description

VARIMAX ROTATION is used to redistribute a factor matrix (principal axis, centroid, etc.) variance so that the matrix approaches orthogonal simple structure. The varimax scheme maximizes the following criterion function:

$$\Sigma_s(h\Sigma_j(a_{(j,s)}{}^{2/h}(j)^2)^2 - (\Sigma_j(a_{(j,s)}{}^{2/h}(j)^2))^2)$$

where j is the variable index number: 1,....., n

s is the factor index number: 1,....., f

$a_{(j,s)}$ is the factor loading of the j^{th} variable on the s^{th} factor

hj^2 is the j^{th} variable communality

For further discussion see:

H. F. Kaiser, "Computer Program for Varimax Rotation in Factor Analysis", Educational and Psychological Measurement, Vol. XIX, Nov. 3, 1959, pp. 413-420.

Cooley and Lohnes, Multivariate Procedures for the Behavioral Sciences, New York, John Wiley and Sons, Inc., 1962, pp. 161-3.

II. Restrictions

The number of factors may be anything greater than or equal to 2. Any factor matrix generated by a statistical system factor analysis program is acceptable input. A matrix may also be entered from cards. If this is the case, the number of rows in the matrix must be specified on the data format card.

III. Parameters

The parameters for the VARIMAX ROTATION appear on the program call card. They must follow the program name in this order:

Parameter Number	Description
1	Input Address. CARDS or SEQUENTIAL 1-15. If CARDS are used the DATA card must contain the number of rows as well as the number of columns in the input matrix (see User's Guide for details).
2	Output Address. SEQUENTIAL 1-15 and/or PRINT.

Parameter Number	Description
3	The presence of a number greater than 0 in this parameter indicates the communalities should be printed.
4	0 or blank for normal VARIMAX. 1 if raw VARIMAX is desired.
5	Output Address of transformation matrix T. SEQUENTIAL 1-15 and/or PRINT.

VARISIM ROTATION

I. General Description

Given an input factor matrix A, this program computes by an iterative procedure an orthonormal transformation matrix T to rotate A to simple structure by the equation

$$AT = B.$$

Thus, the program provides an orthogonal rotation of a factor matrix A of uncorrelated factors to a factor matrix B of uncorrelated factors. The mathematical criterion employed is quite similar to Kaiser's Varimax criterion; however, there exist definite contrasts between the two techniques. First, the Varisim program is considerably slower than the Varimax program, often taking three to four times as long to obtain convergence. The time ratio is dependent on the size of the factor matrix, there being only small differences for small factor matrices. Second, the two methods lead to quite different solutions in the case of a large factor matrix, though results are often quite similar for small matrices. In general, the factors rotated by the Varisim program are characterized by more even contributions to the common test variance than the corresponding Varimax factors. That is, while Varimax attempts to concentrate variance accounted for on the first few factors, Varisim will distribute this variance accounted for more evenly across the factors. Naturally, Varimax, by definition, provides greater simplicity of structure, though usually only slightly superior to Varisim.

II. Restrictions

If input is from cards, the number of rows in the input factor matrix must be specified on the data format card.

III. Parameters

The program mnemonic VSM appears first on the card and is followed by up to 6 parameters.

Parameter Number	Description
1	Input address of factor matrix A. CARDS or SEQUENTIAL 1-15.
2	Output address of rotated factor matrix B. SEQUENTIAL 1-15 and/or PRINT.
3	Output address of transformation matrix T. SEQUENTIAL 1-15 and/or PRINT.
4	Maximum number of iterations; if left blank, 100 iterations will be stopping criterion. (See special comment.)

Parameter Number	Description
5	Exponent of convergence n, where tolerance ETA = 10^{-n} . If left blank, n is set to 3. (See special comment.)
6	Output address of factor matrix A. SEQUENTIAL 1-15 and/or PRINT, or left blank if not desired.

IV. Special Comment

Only in rare situations will the limit of 100 iterations be approached. In most cases, 10 to 20 iterations will be sufficient. The default convergence criterion of 10^{-3} should be adequate in all cases since this has been found to lead to an unambiguous and unique solution. Thus, for virtually any standard input factor matrix, parameters 4 and 5 can be left blank.

TUCKER-MESSICK POINTS OF VIEW ANALYSIS

I. General Comments

One of the classic experimental designs in perceptual research involves subjects making similarity judgments for all possible pairs of a given set of stimuli. The Tucker-Messick model was developed to analyze such data in order to discover whether particular groups of individuals have different viewpoints about stimulus interrelationships. Results of the analysis show what different viewpoints are present within the sample and the extent to which each individual uses each point of view.

Since its conception, however, the technique has been applied to a wide variety of data for purposes of examining individual differences in judgment or performace. This results from the fact that, for any set of measurements for N individuals on n variables, the model specifies the dimensions of greatest variation among individuals, and the extent to which each individual is characterized by each dimension. The technique can be of great value in studying individual differences in many different situations.

II. Description of the Mathematical Model

The present description of the model will be based on a data matrix containing measurements on or by N individuals on n stimuli or variables. (Note: these stimuli can, in fact, be stimulus pairs in a paired-comparison judgment situation.) This program assumes input of an N x n data matrix, X'. The crucial question which the model attempts to answer is whether there exists consistent covariation among groups of individuals on the n variables. The question is resolved by factoring X' into its principal components. The factor solution of X' indicates the number of dimensions required to account for individual differences in performance or judgments. For judgment data, this represents the number of consistent viewpoints being used within the sample. The final result of this procedure consists of one matrix which specifies the dimensions of the stimulus space accounting for greatest individual variation, and another matrix specifying the extent to which each observed individual is characterized by each of these dimensions.

The notation below corresponds to notation in the Tucker-Messick article. Since X is asymmetric and rectangular, it cannot be factored directly. Thus, we use the Eckart-Young procedure to construct a matrix $\hat{X}$ which approximates X but is of lower rank. $\hat{X}$ is constructed from the r largest characteristic roots and vectors of X, according to the Eckart-Young model:

$$(1) \quad \hat{X}_r = U_r \Gamma_r W_r.$$

These components are computed from the cross-products matrix P,

$$(2) \quad P = XX',$$

as follows: analyze P into principal components; i.e.,

$$(3) \quad P = U\Gamma^2U'$$

and truncate to r desired characteristic roots and vectors. We can now solve equation (1) for W_r:

$$(4) \quad W_r = \Gamma_r^{-1}U_r'X.$$

Elements in W_r' represent projections of points corresponding to individuals on the unit-length principal vectors of X. Elements in U_r represent projections of points corresponding to stimuli on the unit length principal vectors of X.

Since each vector of W_r' is composed of N elements and is of unit length, it is obvious that the loadings, or projections of individuals, are dependent on sample size N. We can rescale W_r into a matrix V:

$$(5) \quad V = N^{1/2}W_r.$$

Then the coefficients in V will be independent of sample size. To maintain the Eckart-Young relationship, we must also rescale U_r:

$$(6) \quad Y = U_rN^{-1/2}.$$

Equation (1) can now be rewritten:

$$(7) \quad X_r = Y\Gamma_rV.$$

Elements of V and Y represent scaled projections of individuals and stimuli respectively on the principal vectors of X. Matrix V can then be converted to a factor matrix A of scaled projections of individuals on principal factors by weighting each vector by the square root of the corresponding eigenvalue:

$$(8) \quad A = \Gamma_rV.$$

Combining equations (7) and (8), we find that:

$$(9) \quad \hat{X}_r = YA.$$

This equation represents the final result of the present program.

III. Output and Further Analyses

1. Data matrices computed: This program will compute and print or store on request matrices P, W_r', U_r, Y, V', and A', along with the eigenvalues of P. Printed output includes explanatory labels and equations.

2. Rotation: Y and A might be rotated so that the inherent dimensions of individual variation will be in positions more appropriate for psychological interpretation. A transformation matrix T can be derived to rotate to simple structure, e.g. by the Varimax criterion:

(10) $B = TA$, and

(11) $Z = YT^{-1}$

We then have, combining equations (9), (10), and (11),

(12) $\hat{X}_r = YT^{-1}TA = BZ$

and the Eckart-Young theorem is still satisfied.

3. Person Space Plots: Since entries in matrix B represent coordinates of points for individuals on rotated axes, this space may be readily plotted graphically by making scatter plots for each possible pair of axes. This may be done by hand or by inputting matrix B to the SOUPAC program SCATTER PLOTS. Plots could also be made prior to rotation on the basis of matrix A. These plots can be a crucial step in the analysis of homogeneous subgroups or of widely deviant individuals in the data.

4. Correlating dimensions with outside variables: Since each individual receives a score on all r derived dimensions, correlations may be computed between these dimensions and scores on other outside measures--perhaps personality or performance measures--in order to ascertain properties and correlates of the derived dimensions. This would be accomplished by augmenting the matrix A (or B) with a matrix of measures for the same individuals on other variables and computing a correlation matrix from this data.

5. Idealized Individuals: The user can insert, at any desired location on the plots of the factor space of individuals, additional points which can be interpreted as "idealized individuals." Their location can be determined from any desired criterion, and any number of idealized points can be inserted into the person space. These points of interest may represent centroids of clusters, deviant individuals, etc. By combining the Tucker-Messick program with the MATRIX program, the user can reconstruct raw data for the conceptual individuals based on the r dimensions retained. This would be done as follows:

a) read the coordinates of each ideal point on each factor;
b) record the r coordinates of each point in a row vector;
c) adjoin these row vectors for g idealized individuals into a matrix G';
d) punch the matrix G' and, using the MATRIX program, store it on a sequential file;
e) store the matrix Y as output from the present program;
f) using the MATRIX program, transpose Y to get Y' and compute reconstructed data matrix X_g' via the following multiplication:

$$\hat{X}'_g = G'Y'.$$

This can be done after rotation by substituting rotated dimensions (Z') for unrotated dimensions (Y').

IV. Special Comments

(1) Input: The input matrix is denoted X' and is composed of N rows representing individuals and n columns representing variables or stimuli.

(2) Eckart-Young approximation of X': By outputting matrices U and W' (parameters 6 and 7) along with the eigenvalues of P, the user can obtain the fundamental Eckart-Young resolution of the raw data matrix. By performing the proper matrix manipulations (equation (1)), one can obtain an Eckart-Young approximation to X based on r dimensions.

(3) Specifying the number of factors: The number of dimensions to be retained must be specified in parameter 3. This decision is usually based on a preliminary computer run which computes and factors the matrix P; the number of dimensions retained is determined by the resulting series of eigenvalues.

(4) Type of P-matrix to be used in analysis: Though the mathematical model is written in terms of a cross-products matrix P, this matrix might also be composed of correlations or covariances. This option is specified in parameter 2.

(5) Multi-dimensional scaling: This technique was originally developed to be used with judgment data, in conjunction with multi-dimensional scaling analyses. If the user wishes to proceed in this direction, it is strongly recommended that he have at least a fundamental understanding of the Tucker-Messick model as well as the general concepts involved in multi-dimensional scaling. The SOUPAC office has information about and access to the widely used TORSCA program, commonly used in conjunction with the Tucker-Messick model.

V. Parameters

The program mnemonic VEW appears first on the card and is followed by up to 10 parameters. Since computations are based on parameters which are specified, there should be no blanks in the parameter string. For example, if the user wants output through parameter 9, all previous parameters must be specified. This occurs because the program cannot compute the matrix for parameter 9 without first computing the previous matrices. The exception to this is that parameter 5 can be left blank since eigenvalues will always be computed and printed.

Parameter Number	Description
1	Input address of matrix X'. CARDS or SEQUENTIAL 1-15.
2	Code specifying type of P matrix to be computed: (1) cross-products (2) correlations (3) covariances. (1) is default.
3	Number of factors to be extracted from matrix P.
4	Output address of matrix P. SEQUENTIAL 1-15 and/or PRINT.
5	Output address of row of eigenvalues of P. SEQUENTIAL 1-15. These are printed automatically.
6	Output address of matrix U: projections of stimuli on unit length vectors of X. SEQUENTIAL 1-15 and/or PRINT.
7	Output address of matrix W': projections of individuals on unit length vectors of X. SEQUENTIAL 1-15 and/or PRINT.
8	Output address of matrix V': scaled projections of individuals on principal vectors of X. SEQUENTIAL 1-15 and/or PRINT.
9	Output address of matrix Y: scaled projections of stimuli on principal vectors of X. SEQUENTIAL 1-15 and/or PRINT.
10	Output address of matrix A': scaled projections of individuals on principal factors of X. SEQUENTIAL 1-15 and/or PRINT.

VI. References

Tucker, L. R. and Messick, S. "An Individual Differences Model for Multidimensional Scaling", Psychometrika, 1963, pp. 333-367.

ECONOMETRICS PACKAGE

ECONOMETRIC REDUCED FORM AND RESIDUAL ANALYSIS

I. General Description

The ECONOMETRIC REDUCED FORM AND RESIDUAL ANALYSIS program performs operations on the model $X\beta + Y\Gamma = U$. Using the following definitions (dimensions of matrices are given in parentheses):

T is the number of observations
NY is the number of equations in the model and the number of jointly dependent variables in the model since the two must be the same
NX is the number of predetermined variables plus the constant term
N is the number of variables plus the constant term; N = NY + NX
$[XY]_{(T,N)}$ is the raw data matrix plus a column of constant terms

$\begin{bmatrix}\hat{\beta}\\ \hat{\Gamma}\end{bmatrix}_{(N,NY)}$ is the matrix of coefficient estimates. This matrix includes the estimate of the intercept term

The program calculates the following:

(1) Estimate of residuals: $\hat{U}_{(T,NY)} = (XY)_{(T,N)} \begin{bmatrix}\hat{\beta}\\ \hat{\Gamma}\end{bmatrix}_{(N,NY)}$

(2) Durbin-Watson statistic for each equation (i):

$$DW(i) = \frac{\sum_{t=2}^{T} [\hat{U}_i(t) - \hat{U}_i(t-1)]^2}{\sum_{t=1}^{T} [\hat{U}_i(t)]^2}$$

(3) Estimate of the variance-covariance matrix of residuals:

$$\hat{\Sigma}_{(NY,NY)} = \frac{1}{T} \hat{U}'_{(NY,T)} \hat{U}_{(T,NY)}$$

(4) Reduced form estimates: $\hat{\Pi}_{(NX,NY)} = (X'X)^{-1}_{(NX,NX)} (X'Y)_{(NX,NY)}$

(5) Reduced form predicted values: $\tilde{Y}_{(T,NY)} = X_{(T,NX)} \hat{\Pi}_{(NX,NY)}$

(6) Estimate of reduced form residuals: $\hat{V}_{(T,NY)} = Y_{(T,NY)} - \tilde{Y}_{(T,NY)}$

(7) Estimate of the variance-covariance matrix of reduced form residuals:

$$\hat{\Omega}_{(NY,NY)} = \frac{1}{T} \hat{V}'_{(NY,T)} \hat{V}_{(T,NY)}$$

II. Restrictions

Only those inputs used in the calculations called for need be given. They must be in the following formats:

(1) Coefficients:

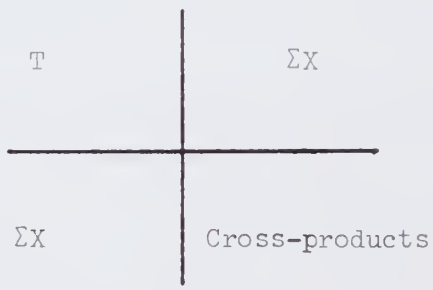
The coefficient matrix for K equations with N-1 variables, N1 predetermined and N2 jointly dependent, must be a K by N matrix. Each row corresponds to an equation. The first element in each row is the constant term followed by the coefficients matrix (i.e., predetermined coefficients first; jointly dependent coefficients next). In each row, there must be -1 which corresponds to the jointly dependent variable that was normalized on.

(2) Raw Data:

The data must be arranged so that predetermined variables occur first and jointly dependent variables last. (The TRANSFORMATION program may be used to arrange data in this way, if it is not already like this).

(3) Raw Data Cross-products Matrix:

The cross-products matrix must have the following form:



Care should be taken to see that an input address is specified for any data needed in calculating the desired statistics and that any intermediate statistics needed are stored(i.e., an output address besides print is specified). The following list indicates which previous statistics are needed in the calculation of each statistic.

1. Estimate of Residuals - coefficients and raw data
2. Durbin-Watson statistic - coefficients and raw data
3. Estimate of Covariance matrix of residuals - coefficients, raw data and the number of observations
4. Estimated Reduced form coefficients - raw data
5. Reduced form predicted values and residuals - reduced form coefficients (no output address) and raw data

(6) Estimate of Covariance matrix of reduced form residuals--estimate of reduced form residuals and the number of observations

II. Parameters

The parameters appear on the program card following the mnemonic ECON in the following order (See also Special Comments):

Parameter Number	Description
1	Input Address for coefficients. SEQUENTIAL 1-15. Same as output address for K-CLAS.
2	Input Address for raw data cross-products matrix. SEQUENTIAL 1-15. Same as output address for K-CLAS.
3	Input Address for raw data. SEQUENTIAL 1-15. (See Special Comments).
4	Number of predetermined variables (total).
5	Output Address for estimates of residuals. SEQUENTIAL 1-15 and/or PRINT.
6	If P, the estimate of the variance-covariance matrix of residuals is printed.
7	If P, (estimated) reduced forms are calculated and printed.
8	If P, reduced form predicted values and residuals are printed.
9	If P, the estimate of the variance-covariance matrix of reduced form residuals is printed.

IV. Special Comments

If Parameter Number 3 is specified, the Durbin-Watson statistic will be calculated and printed.

The ECONOMETRIC REDUCED FORM AND RESIDUAL ANALYSIS program requires input from the K-CLASS ESTIMATION program and/or the THREE STAGE LEAST SQUARES program. The following example illustrates.

V. Example

```
/*ID       [accounting information to include REGION size]
//  EXEC  SOUPAC
//SYSIN  DD  *
TRA(C).
MOV(C)(S4).
END P
K-C(S4)(S2)*1*(7)(S1)(S3)(P).
(3)(1)(1)(2)(3)(8).
(3)(1)(1)(2)(4)(10).
(3)(1)(5)(6)(7)(9).
END P
ECON(S2)(S3)(S4)(7)(S5/P)(P)(P)(P)(P).
END S
```

Notice that there is no ENDP card after the ECON program because ECON has only a main parameter card.

VI. References

Johnston, J., Econometric Methods, New York, McGraw-Hill Book Company,Inc. 1960.

Goldberger, Arthur S., Econometric Theory, New York John Wiley and Sons, Inc., 1964.

K-CLASS ESTIMATION

I. General Description

There are three estimators which belong to the K-class. These include: Ordinary Least Squares (multiple regression), Two-stage Least Squares, and Limited Information Maximum Likelihood.

II. Description of K-Class Output

The K-Class program calculates the basic statistics:

Mean: $\bar{X}_i = \frac{\Sigma X_i}{N}$ where N = Sample Size

Variance Covariance: $S_{ij} = \frac{\Sigma X_i X_j}{N} - \bar{X}_i \bar{X}_j$

Standard Deviation: $s_i = \sqrt{S_{ii}}$

Correlation: $C_{ij} = \frac{S_{ij}}{s_i s_j}$

Cross-products in matrix notation: $CP = X'X$

K-Class also calculates the eigenvalue to be used in Limited Information Maximum Likelihood estimation.

K-Class then goes on to calculate estimates and associated statistics for Ordinary Least Squares (OLS), Limited Information Maximum Likelihood, and Two-Stage Least Squares.

Formulas

$\hat{\beta}'$ $\equiv$ estimates of the jointly dependent coefficients

$\hat{\gamma}$ $\equiv$ estimates of the predetermined coefficients

y_o $\equiv$ variable normalized on

Y $\equiv$ jointly dependent variables in the equation

X_* $\equiv$ predetermined variables in the equation

X $\equiv$ predetermined variables in the system

a $\equiv$ intercept term

e $\equiv$ error term

Estimating Formulas

$$\begin{bmatrix} \hat{\beta} \\ \hat{\gamma} \end{bmatrix} = - \begin{bmatrix} Y'Y - kV'V & Y'X_* \\ X_*'Y & X_*'X_* \end{bmatrix}^{-1} \begin{bmatrix} Y'y_o - kV'y_o \\ X_*'y_o \end{bmatrix}$$

or

$$\hat{\Theta} = A^{-1} J$$

where $V'V = Y'Y - Y'X(X'X)^{-1}X'Y$

$$V'y_o = Y'y_o - Y'X(X'X)^{-1}X'y_o$$

k determines the estimating technique and is an arbitrary scalar which may be either random or nonstochastic.

Standard Error of Estimate

$$\hat{\sigma} = \sqrt{\frac{y_o'y_o - (\hat{\beta}\ \hat{\gamma})\ C}{N-j}}$$

where j = Rank of A^{-1}

Standard Error of the Estimated Coefficients.

s^*_{ii} ≡ The square root of the i^{th} diagonal element of the

$\hat{\sigma}^2 A^{-1}$ matrix

T-Ratio:

$$t = \frac{\hat{\Theta}_i}{s^*_{ii}}$$

Covariance matrix of the coefficients:

$$C^2 = \hat{\sigma}^2 A^{-1}$$

Intercept term :

$$a = y_o - Y(\hat{\beta}) - (X_*)(\hat{\gamma})$$

II. Ordinary Least Squares (OLS)

When k = 0 is specified, Ordinary Least Squares estimates are computed. Y is then assumed to be another predetermined variable in the equation. When ordinary least squares is specified the following additional statistics are supplied.

In matrix notation

$$R^2 = \frac{\hat{\theta}C - \Sigma y_o^2 / N}{y_o'y_o - \Sigma y_o^2 / N} = \frac{\text{Explained Sum of Squares}}{\text{Total Sum of Squares}}$$

Total Sum of Squares: $TSS = y_o'y_o - (\Sigma y_o)^2/N$

Regression (Explained) Sum of Squares: $RSS = \hat{\theta}C - \Sigma (y_o^2)/N$

Error (Unexplained) Sum of Squares: $ESS = TSS - RSS = y_o'y_o - \hat{\theta}C$

Example: Suppose that there were two possible models that one wanted to estimate. In one model variable 10 is included and in the other variable 10 was not included.

```
K-C(S1)(S4)*0*(10)(S2)(S3)(P).
(10)(1)(1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11).
(9)(1)(1)(2)(3)(4)(5)(6)(7)(8)(9)(11).
END P
```

IV. Limited Information Maximum Likelihood (LIML)

When k = μ, where μ is the smallest eigenvalue of the equation, is specified, then Limited Information Maximum Likelihood estimates are computed.

The eigenvalue is calculated in the following manner:

Let $W_* = Y'Y - Y'X_* (X'X)^{-1} X_*' Y$

$W = Y'Y - Y'X(X'X)^{-1} X'Y$

K-Class uses the smallest eigenvalue of the matrix $W^{-1}W_*$.

Example: Suppose that there is a two equation system with four predetermined and two jointly dependent variables. The following program will calculate LIML estimates for both equations:

```
K-C(C)(S3)*-1*(4)(S1)(S2)(P)(S5).
(2)(2)(1)(4)(5)(6).
(2)(2)(2)(3)(5)(6).
END P
```

V. Two-Stage Least Squares (2SLS)

When k = 1 is specified Two-Stage Least Squares estimates are computed.

Example: Suppose one has a three equation model that contains three jointly dependent and ten predetermined variables. The following program will calculate 2SLS estimates.

```
K-C(C)(S3)*1*(10)(S1)(S2)(P).
(5)(2)(1)(2)(3)(4)(5)(11)(13).
(4)(2)(3)(4)(6)(7)(12)(13).
(5)(2)(1)(2)(8)(9)(10)(11)(12).
END P
```

VI. Parameters

A. Main Parameters

Parameter Number	Description
1	Input Address. CARDS or SEQUENTIAL 1-15.
2	Output Address for estimated coefficient. SEQUENTIAL 1-15. Same as input address for ECON.
3	Floating point value of k. (See special comments) This value should be enclosed in asterisks.
4	Number of predetermined variables in the system.
5	Output address for cross-products matrix. SEQUENTIAL 1-15 and/or PRINT. Same as input address for ECON.
6	Output address for raw data covariance matrix. SEQUENTIAL 1-15 and/or PRINT.
7	If P, correlations matrix will be printed.
8	Output address of eigenvalues if LIML. SEQUENTIAL 1-15.
9	Type of Input 0 = raw data 1 = cross-products 2 = covariance If the input is raw data the ninth parameter may be omitted.

B. Sub-parameters (Equation Control Cards)

Subparameters are needed for all K-Class programs. There is one sub-parameter card for each equation. Each equation card has the following form:

Parameter Number	Description
1	Number of predetermined variables in the equation.
2	Number of jointly dependent variables in the equation.
3	The variable number of all variables in the equation in the order: 1 - predetermined in the equation 2 - jointly dependent in the equation with variable standardized on last.

C. DATA cards must be punched with predetermined variables first, jointly dependent variables last.

VII. Special Comments

A. If k = *0* Ordinary Least Squares Estimates are computed.
If k = *1* 2-Stage Least Squares Estimates are computed.
If k = *-1* Limited Information Maximum Likelihood Estimations are computed.

B. K-Class accepts data from cards or intermediate storage either as raw data, cross-products, or covariance. If cross-products or covariance are used as input to K-Class, the matrix must be in the following order:

Cross-products

N	Σx
Σx	cross-products

Covariance

N	$\bar{X}$
σ	covariance

VIII. References

Goldberger, A.S., Econometric Theory, New York, John Wiley and Sons, Inc., 1964.

Johnston, J., Econometric Methods, New York, McGraw-Hill Book Company, Inc., 1960.

LINEAR PROGRAMMING

General Description

LINEAR PROGRAMMING maximizes or minimizes a linear function subject to certain linear inequalities called constraints.

In matrix notation:

Find the solution to

$AX <, =, > b$ (a system of linear equations or inequalities)

which maximizes (or minimizes)

$Z = CX$

where $X \geq 0$

A is the matrix of coefficients of the constraints, X the vector of variables, C the vector of costs or profits associated with each variable, and b a vector or matrix of non-negative constants which places a bound on the linear equations.

The equations, $AX <, =, > b$ in n variables define and bound a space called the feasible space in which all allowable values of the n variables are defined. The SIMPLEX criterion finds those combinations of variables which optimize the objective function within this feasible space. To solve the system of linear equations defined above, the inequalities must be changed to equalities. This is accomplished by addition of surplus variables to "greater than" constraints, and slack variables to "less than" constraints. To create the basis for solving a system of linear equations, an identity matrix must be formed and augmented to the A matrix of structural variables. Creation of the identity matrix is completed by addition of artificial variables to constraints with a "greater than" relational operator. The program adds any needed variables.

Since there are more variables (structural + surplus + slack + artificial) than rows, some method must select which variables will be in solution. The SIMPLEX Algorithm selects a number of variables (equal to the number of rows) which will be in solution. The final solution is the maximum (or minimum) of the linear function subject to the constraints. Since slack and surplus variables have "real" meaning, they may appear in the final and intermediate solutions. Their presence as a non-zero value indicates that the constraint to which they were added is not binding. Artificial variables have no "real" meaning. Presence of artificial variables in solution indicates that some constraints are so constructed as to preclude a solution which has "real" meaning.

The slack and surplus variables are given costs of zero in the objective function. Artificial variables are given large negative costs. SIMPLEX attempts to drive artificial variables from solution.

Failure to drive artificial variables from solution may indicate a problem in which constraints are mutually exclusive or that the cost assigned to the artificial variable is not large enough.

In matrix notation the augmented matrix before calculations begin would appear as:

$$A \mid I \mid S \mid = b$$

where I is the identity matrix of slack and artificial variables and S is the matrix of surplus variables. Row operations are performed on the augmented matrix according to the SIMPLEX criterion. After any number of row operations, the inverse matrix of the original coefficients of structural variables now in solution is contained in the columns where the original identity matrix was located. At every stage (row operation) an identity matrix will be present. This identity matrix indicates the variables in solution.

Since the original table is stored by the program, it is possible to compare the results of the inverse obtained through LINEAR PROGRAMMING with the inverse obtained by a standard inversion technique. The user may set the absolu value for this comparison in Parameter 3. If the comparison does not meet the accuracy requirement, a new table is formed using the original table and the calculated inverse. After a feasibility check, the program continues calculatio until an acceptable solution is obtained.

References

Llewellyn, R. LINEAR PROGRAMMING. New York, New York: Holt, Rinehart, and Winston, 1966.

Hadley, G. LINEAR PROGRAMMING. Reading Massachusetts: Addison-Wesley, 1963.

II. Restrictions

The program is limited to a maximum of 90 rows or constraints, 300 columns or variables, and 5 columns in the requirement matrix.

These limits are internal limits and the user is warned that large problems may exceed the program capacity during accuracy check and calculations involving multiple column requirement matrices. Program capacity WILL be exceeded if: the number of constraints + number of structural variables + number of "greater than" inequalities > 300.

Input may come ONLY from CARDS in the form of subparameters.

III. Parameters

All floating point numbers (indicated by FP) must be enclosed by a pair of asterisks. All integer numbers (indicated by IN) must be enclosed in parentheses The main call to the program and each subparameter must be terminated by a period (.).

The program is entered by punching the symbols L-P followed by the appropriate main parameters and subparameters. All main parameters have default options.

Main Parameters to follow L-P

1 Cost of artificial variables *large negative FP numbers*. Default = -1.E50.

2 Minimum value for calculations *FP*. If any calculation falls below this value, it is set to zero. Default = internal calculations.

3 Value for accuracy check *FP*. If absolute value for calculated difference (See General Description) falls below this value, final value is termed inaccurate and calculations are performed to correct rounding errors. Default = .5 .

4 If 1, suppress print of solution matrix (IN).

5 If 1, suppress print of check matrix (IN).

6 Print every INth step, i.e. row operation (IN).

7 If 1, insert small positive, non-zero number for any zero in the b vector. Useful aid if b vector contains many zeros.

Subparameter

The program now expects to find the word MINimize or MAXimize followed by a string of constants which represent, in sequential order, the cost or values associated with each variable. All non-zero constants (with or without decimal) must be enclosed by a pair of asterisks. Zeros may be enclosed by asterisks. A series of sequential zeros may be represented by a pair of parentheses, i.e. the integer number in the pair of parentheses represents the number of sequential zeros to be inserted. All coefficients must appear and be in sequential order.

The cost coefficients representing the objective function are terminated by a period. The constraints are entered in a similar manner. All variables must be in sequence. Coefficients of zero must be included. Multiple requirement vectors are entered in the standard form. The constraint is terminated by a period. Comments which do not include period (.), comma (,), asterisks (*) or left parenthesis may be entered at any point outside those characters delimiting constants. The requirement vectors are separated from the rest of the constraint by relational operators. All coefficients must appear and be in sequential order.

The program recognizes three relational operators: LE (less than or equal), EQ (equal), and GE (greater than or equal). These relational operators are surrounded by quotes ("). See Section V. Examples in this program.

Output

The output consists of the objective function, the final solution matrix, the variables in solution, and the optimal functional value. In addition, Shadow Prices or opportunity costs are printed. Shadow Prices provide useful information on the "cost" of having certain constraints, or the increased profit to be obtained by 'relaxing' a particular constraint.

For example:

Constraint 1: $1X(1) + 2X(3) \leq 5.0$

To this constraint, slack variable X(I) is added to make it an equality. In the final solution, X(I) is not in solution. The optimal maximum functional value is 20. The 'Shadow Price' on variable X(I) is 2.0. This means that if we relax this constraint to 6.0, the optimal maximum value could be 22.0. For every unit the constraint is relaxed, the functional value will be changed by the Shadow Price. The Shadow Price holds until the constraint is no longer binding. The same logic may be applied to "GE" type constraints with surplus variables. For interpretation of Shadow Price for structural and artificial variables, the user is referred to texts under headings such as "Dual Algorith Interpretation of the Dual", and "Opportunity Costs".

Basis variables refer to those variables which form the original identity matrix. The variable numbers are listed in the order they were added to the constraints. The number of basis variables will always equal the number of constraints. To determine whether a basis variable is a slack or artificial variable, refer to the coefficients of these variables in the objective function. A slack variable will have a coefficient of 0.0.

MESSAGES

PROBLEM TOO LARGE: More than 300 variables or 100 constraints on input or during addition of slack, surplus, and artificial variables.

NORM FOR CUTOFF: Value of Main Parameter Number 2, either supplied or default.

ERROR IN SIMPLX: Source Program Error. See a consultant.

SOLUTION UNBOUNDED: Constraints do not form a closed space. Optimal functional value is infinite.

NUMBER OF ITERATIONS: Number of row operations needed to calculate final solution. For multiple requirement vectors, number is not cumulative.

ACCURACY ACCEPTABLE or ACCURACY NOT ACCEPTABLE: Comparison with Main Parameter Number 3.

VARIABLE ADDED

NEW BASIS VARIABLES ARE: Iterations either inaccurate and new variable added or, during execution of multiple requirement vector, a new variable had to be added to make problem feasible (requirement vector positive).

NON-RESOLVABLE TIE: Cannot occur mathematically. Only reason for occurance is due to rounding error in machine. Can be corrected by incrementing or decrementing requirement vector by a small amount. Perform this only for constants of same value. (Use Parameter 7).

Other messages should be self-explanatory.

IV. Special Comments

Speed and accuracy can be increased by observing the following suggestions

1) Never make Parameter 3 (accuracy check) larger than (0.1) X (number of significant digits in table). For example, if numbers in the table are 4, 5, .001, 86, 95.32, you have "one" significant digit. Set Parameter 3 to *.1*.

2) Scale numbers in table to get them into same range. For example, if table entries are of the order 10^1, and the requirement vectors are of the order 10^3, scale requirement vectors to 10^1 and rescale solution by 10^2. The objective function may also be rescaled in a similar manner. Rescaling essentially reflects the number of significant digits.

Examples

The problem:

Minimize -.75X(1) + 150X(2) -.02X(3) + 6X(4)

Subject to the following constraints:

Constraint(1)

.25X(1) -60X(2) -.04X(3) + 9X(4) $\leq$ 0, 1, 2

Constraint(2)

.05X(1) -90X(2) -.02X(3) -3X(4) $\leq$ 0, 1, 2

Constraint(3)

1X(3) $\leq$ 1, 2, 3

Could be set up on cards as follows:

```
/*ID
//  EXEC  SOUPAC
//SOUPAC.SYSIN  DD  *
L-P*-1.E20****.1*()()(1).
MIN*-.75**150**-.02**6*.
LABOR *.25**-60**-.04**9*"LE"*0**1**2*.
LAND*.05**-90**-.02**-3.0*"LE"*0**1**2*.
CASH(2)*1*(1)"LE"*1**2**3*.
END PROGRAM
END SOUPAC
/*
```

This problem will result in an unbounded solution with requirement vector number one. The problem terminates without performing calculations on the other vectors.

Note insertion of sequential zeros on CASH card.

QUADRATIC PROGRAMMING

I. General Description

This program maximizes the quadratic function $cX + 1/2\ X^TDX$ subject to the linear constraints $AX \leq b$, where c is an n-vector, D is a symmetric negative definite n by n matrix, A is an m by n matrix of coefficients or constraints and b is an m vector.

The Kuhn-Tucker theory shows that a solution to the constrained maximization problem is obtained if and only if vectors X, L, V, and W can be found such that:

$$DX - A^TL + V = -c$$

$$AX + W = b$$

where the elements of X, L, W, and V are non-negative and the conditions $V^TX=0$ and $W^TL=0$ are satisfied. To find these vectors, artificial vectors Z^1 and Z^2 are added to the first equation and a y-vector is added to the second. Simple techniques are then used to eliminate first the Y and then the Z variables.

References:

Carr, C. R. and C. H. Howe, Quantitative Decision Procedures in Management and Economics, McGraw-Hill, 1964.

Hadley, G., Nonlinear and Dynamic Programming, Addison-Wesley, 1964.

Wolfe, P., "The Simplex Method for Quadratic Programming", Econometrica, 27, 1959, pp. 382-398.

NOTE: Carr and Howe claim that elements of the W-vector may not be entered in the first stage of the simplex procedure. Since this requires that there exist a solution to AX = b, it is a severe restriction. It is also unnecessary, and this program does enter W-variables during the first stage. Otherwise, the procedures used closely follow those of Carr and Howe.

II. Restrictions

The maximum number of X-variables is 40. The number of x-variables plus the number of constraints must be $\leq$ 80.

The D-matrix must be negative definite. If this is doubtful, use the PRINCIPLE AXIS FACTOR ANALYSIS program to extract the eigenvalues. All must be negative. Semi-definite D-matrices may be perturbed or the user may limit the number of iterations to be performed. If this limit is exhausted, final solution vectors will be printed out (see below).

The only form of input is a matrix of data. If there are n x-variables and m constraints, the matrix should have n + 1 columns and m + n rows, partitioned as follows:

D (n x n)	c (n x 1)
A (m x n)	b (m x 1)

Note that this is the c vector, not the -c vector mentioned in the Kuhn-Tucker formulas. Also note the + sign and the 1/2 coefficient of the xDx term. All constraints in this type of input are assumed to be $\leq$ type. Multiply $\geq$ constraints through by -1. The equality constraing:

$$\sum_{j=1}^{n} a_{ij}X_j = b_i$$

is equivalent to the two constraints $\Sigma a_{ij}X_j \leq b_i$ and $\Sigma -a_{ij}X_j \leq b_i$.

This matrix can be read in from cards or from temporary storage.

The elements of the w-vector are always non-negative and are to be considered "slack" for $\leq$ constraints and "surplus" for $\geq$ constraints.

The user may obtain the basis vector at the end of each iteration showing which variables are in the basis and their quantities (option 2). He may alternatively have the entire matrix printed out after each iteration (option 3). The user is cautioned that option 3 can use immense quantities of paper and time unless the problem is very small.

III. Parameters

The program call card should have the name QUADRATIC PROGRAMMING followed by these parameters:

Parameter Number	Use or Meaning
1	Input Address. SEQUENTIAL 1-15 or CARDS.
2	Number of Contraints.
3	Output option: 0 if final results only 1 if iterated basis vectors 2 for entire iterated matrix
4	Limit on number of iterations if desired. Leave blank otherwise. Default is 1000.

Parameter Number	Use or Meaning
5	Pertubation quantity. Punch quantity to be subtracted from diagonal of D-matrix between asterisks instead of parenthesis; e.g., *.001*. Leave blank if not desired.

IV. Examples

Example I:

Suppose we wish to maximize the quadratic function

$$F = 10x_1 + 20x_2 + 15x_3 - 1x_1^2 - 2x_2^2 + 1x_1x_2$$

subject to the constraints

$$\begin{aligned} 2x_1 + 3x_2 + 1x_3 &\leq 50 \\ 1x_1 \qquad + 3x_3 &\leq 70 \\ 3x_1 + 2x_2 \qquad &\leq 60 \end{aligned}$$

Since the D-matrix is only negative semi-definite, it should be perturbed to insure convergence to a solution. The following set of cards would solve the problem using data matrix input:

```
/*ID
// EXEC SOUPAC
//SYSIN  DD  *
QUADRATIC PROGRAMMING (CARDS)(3)(0)(0)*.001*.
END SOUPAC
DATA(4)(4F3.0)
-2  1  0 | 10
 1 -4  0 | 20    c
 0  0  0 | 15
---------+----
 2  3  1 | 50
 1  0  3 | 70    b
 3  2  0 | 60
END#
/*
```

D

A

Example II:

Maximize

$$F = 8x_1 + 10x_2 - x_1^2 - x_2^2$$

subject to the constraint

$$3x_1 + 2x_2 \leq 6$$

The D-matrix is negative definite. The problem would be set up as follows:

```
/*ID
//  EXEC  SOUPAC
//SYSIN  DD  *
QUAD (C)(1)(2).
END SOUPAC
DATA(3)(F2.0,2F3.0)
D -2  0 | 8     c
   0 -2 | 10
---------------
A  3  2 | 6     b
END#    |
/*
```

The extreme value of the objective function for this example is .213 E02 .

EXACT RESTRICTED LEAST SQUARES

I. General Description

Exact restricted least squares can be used in two ways: 1) to include prior information about a parameter, or 2) to test a linear hypothesis.

Assumptions:

1) $Y = X\beta + u$

2) $u \sim N(o, \sigma^2 I)$

3) X is nonstochastic

4) X has rank $K < T$

5) $r = R\beta$ r is a j x 1 known vector, R is a j x K known matrix and β is a K x 1 vector of parameters in the model.

Assumption 5) is the hypothesis to be tested. Exact restricted least squares minimizes $(Y - X\hat{\beta}_R)'(Y-X\hat{\beta}_R)$ subject to $r = R\beta$, where $\hat{\beta}_R$, the restricted estimator, is given by

$$\hat{\beta}_R = \hat{\beta} + (X'X)^{-1}R'[R(X'X)^{-1}R']^{-1}(r-R\hat{\beta}), \text{ and}$$

$\hat{\beta}$ is the ordinary least squares estimator.

$$\text{Var}(\hat{\beta}_R) = \sigma^2[(X'X)^{-1} - (X-X)^{-1}R'[R(X'X)^{-1}R']^{-1}R(X'X)^{-1}]$$

An unbiased estimate of σ^2 given that the prior information is true, is

$$S_R^2 = \frac{(Y-X\hat{\beta}_R)'(Y-X\hat{\beta}_R)}{T - K + j} = \frac{S\ SE_{\hat{\beta}R}}{T-K+j}$$

The F test that is given to test the null hypothesis that $r = R\beta$. The F test is calculated:

$$F' = \frac{\dfrac{SSE_{\hat{\beta}_R} - SSE_{\hat{\beta}}}{j}}{\dfrac{SSE_{\hat{\beta}}}{T-K}}$$

$$F' \sim F_{(j,T-K)}$$

II. Parameters

The following parameters should follow the mnemonic RLS:

Parameter Number	Description
1	Input address for raw data. SEQUENTIAL 1-15 and CARDS.
2	Input Restriction Matrix. SEQUENTIAL 1-15 and CARDS.
3	Number of restrictions.
4	Number of equations.
5	1 if want cross products matrix printed.
6	1 if want covariance matrix printed.
7	1 if want covariance matrix of ordinary least squares estimates.
8	1 if want correlation matrix printed.

III. Subparameters - equation control cards

1	Number of exogenous variables in the equation.
2 - K+1	Variable number of the K exogenous variables.
K+2	Variable number of the endogenous variable.

The intercept term is referred to as the coefficient of variable 0. Variable 0 is considered to be an exogenous variable hose value is always 1.

IV. Input

If all input is from cards, the decks are read in the following sequences:

1) Raw data
2) Restrictions matrix which must be in the form $[R:r]_{j \times (K+1)}$

V. Output

The program prints out ordinary least squares estimates first as part of an intermediate step. Also, the F test and a residual covariance matrix are calculated and printed out under ordinary least squares estimation. Coefficients, errors, and the F test are calculated under the restrictions and printed out.

VI. Sample Program

Suppose we are estimating a general linear model of the form

$$Y = \beta_o + \beta_1 X_1 + \beta_2 X_2$$

where β_o is the intercept term and β_1 and β_2 are the coefficients of the exogenous variables. Let us esimate the coefficients by ordinary least squares and test the hypothesis that the exogenous coefficients sum to 1. The Restrictiom matrix would be of the form:

$$r = R\beta =$$

$$1 = [0\ 1\ 1]\begin{bmatrix} \beta_o \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

The following sample program describes the parameters and the order of the cards needed to obtain exact restricted least squares for this model.

```
/*ID
//  EXEC  SOUPAC
//SYSIN  DD *
RLS(C)(C)(1)(1)(1)(1)(1).
(3)(0)(1)(2)(3).
END P
END S
DATA(3)(3F10.0)
  :
  :   data deck
  :   [X1 X2 Y]
  :
END#
DATA(4)(4F5.0)
0.     1.     1.     1.     [R:r]
END#
/*
```

Note: Variable 0 is not included in the raw data matrix; but its coefficient, β_o, is included in the restriction matrix.

VII. References

[1] Goldberger, Arthur S., Econometric Theory, New York, John Wiley and Sons, Inc., 1960.

[2] Judge, G. G., and Yancey, T. A., "The Use of Prior Information in Estimation of the Parameters of Economic Relationships," Metroeconomics, Vol. XXI (1967).

STOCHASTIC RESTRICTED LEAST SQUARES

I. General Description

Stochastic restricted least squares can be viewed as a special case of generalized least squares. Usually we have some prior knowledge of the approximate size of the coefficients. It seems reasonable that by using this information we can obtain more efficient estimates.

Assumptions:

1) $Y = XB + u$

2) $u \sim (0, \sigma^2 I)$

3) X is a set of fixed variates

4) X has rank $K < T$

5) $r = RB + v$ r is a j x 1 known vector and R is a j x k known matrix of prior information with the stochastic term v.

6) $v \sim (0, \Psi)$

We can develop the model in the following manner:

(1) $\begin{bmatrix} Y \\ r \end{bmatrix} = \begin{bmatrix} X \\ R \end{bmatrix} B + \begin{bmatrix} u \\ v \end{bmatrix}$

where $E\begin{bmatrix} u \\ v \end{bmatrix}[u'v'] = \begin{bmatrix} \sigma^2 I & 0 \\ 0 & \Psi \end{bmatrix} = \Phi$

Apply generalized least square to equation (1)

$$B^* = [(X'R')\Phi^{-1}\begin{bmatrix} X \\ R \end{bmatrix}]\ X'R'\Phi^{-1}\begin{bmatrix} Y \\ r \end{bmatrix}$$

which reduces to

$$B^* = [\frac{X'X}{\sigma^2} + R'\Psi^{-1}R]^{-1}\ [\frac{X'Y}{\sigma^2} + R'\Psi^{-1}r]$$

Since σ^2 is usually unknown Theil suggests that a consistent estimate of σ^2 be used to replace it.

Use $s^2 = \frac{(Y-X\hat{B})'(Y-X\hat{B})}{T-K}$ where the $\hat{B}$'s are the ordinary least squares estimates of B. This new estimator

$$\hat{B}^* = \left[\frac{X'X}{s^2} + R'\Psi^{-1}R\right]^{-1} \left[\frac{X'Y}{s^2} + R'\Psi^{-1}r\right]$$

will be consistent and have the same asymptotic moment matrix as B*.

Using additional information is not enough. We would like to know if the prior information is compatible with the sample information. Theil develops a compatability statistic

$$\delta = (r-R\hat{B})' \left[\sigma^2 R(X'X)^{-1}R' + \Psi\right]^{-1} (r-R\hat{B})$$

which is distributed as Chi-square with j degrees of freedom. This statistic tests the hypothesis that the sample and prior information are compatible. Since σ^2 is usually unknown we must substitute s^2 into the statistic

$$\hat{\delta} = (r-R\hat{B})' \left[s^2 R(X'X)^{-1}R' + \Psi\right]^{-1} (r-R\hat{B})$$

which will have the same asymptotic distribution as δ. Judge and Yancey [1] develop an alternative feasible compatability statistic:

$$\hat{\delta}^* = \frac{\hat{\delta}}{j}$$

They showed that $\hat{\delta}^* \sim F_{(j,\ T-K)}$

The program will print out Theil's compatability statistic $\hat{\delta}$. If the user desires $\hat{\delta}^*$, just divide Theil's statistic by j.

II. Parameters

The following parameters should follow the mnemonic STO

Parameter Number	Description
1	Input Address for raw data. Sequential 1-15 and CARDS
2	Input Restrictions Matrix. Sequential 1-15 and CARDS
3	Number of restrictions
4	Number of equations
5	Input address of covariance matrix of prior information: Ψ. Sequential 1-15 and CARDS
6	1 if want cross products matrix printed

Parameter Number	Description
7	1 if want covariance matrix printed
8	1 if want correlation matrix printed

II. Subparameters - equation control cards

Parameter Number	Description
1	Number of exogenous variables in the equation
2-K+1	Variable numbers of the K exogenous variables
K+2	Variable number of the endogenous variable

The intercept term is referred to as the coefficient of variable 0. Variable 0 is considered to be an exogenous variable whose value is always 1.

IV. Input

If all input is from cards, the decks are read in the following sequence:

1) Raw Data
2) Restrictions Matrix which must be in the form $[R:r]_{j \times (K+1)}$
3) Covariance Matrix of the restriction

V. Output

Ordinary Least Squares and the corresponding F test are printed out as an intermediate step. Theil's compatibility statistic and the stochastically restricted coefficients and errors are also printed out.

VI. Sample Program

Suppose we have the model

$$Y = X\begin{bmatrix}\beta_0\\ \beta_1\\ \beta_2\\ \beta_3\\ \beta_4\end{bmatrix} + u$$

In addition to this simple linear model we believe that $\sum_{i=1}^{4} B_i = 1$. Since we don't know this with complete certainty, we assign a variance of 1/16 to this prior knowledge.

In matrix form then

$r = RB + v$ becomes

$$1 = [01111] \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix} + v$$

and $v \sim (0, 1/16)$

The following sample program describes the parameters and order of the cards needed to obtain stochastic restricted least squares estimates for this model.

```
/*ID
// EXEC SOUP
//SYSIN DD   *
STO(C)(C)(1)(1)(C)(1)(1)(1).
(5)(0)(1)(2)(3)(4)(5).
END P
END S
DATA(5)(5F10.0)
    :                [X1...X4Y]
    :  data deck
    :
    :
END#
DATA(6)(6F5.0)
0.    1.    1.    1.    1.    1.     [R:r]
END#
DATA(1)(F10.0)
 .0625       [Ψ = .0625]
END#
/*
```

Note: Variable 0 is not included in the raw data; but its coefficient, β_0, is considered to be in the restriction matrix.

VII. References

[1] Judge, G. G., and Yancey, T. A., "The Use of Prior Information in Estimation of the Parameters of Economic Relationships," Metroeconomics, Vol. XXI (1969).

[2] Theil, H., "On the Use of Incomplete Prior Information in Regression Analysis, "Journal of American Statistical Association, Vol. 58 (1963).

[3] Theil, H., and Goldberger, A. S., "On Pure and Mixed Statistical Estimation in Economics," International Economic Review, Vol. 2 (1961).

[4] Yancey, T. A., Judge, G. G., and Bock, M.E., "A Mean Square Error Test When Stochastic Restrictions Are Used in Regression," Quantitative Economics Workshop Paper, Department of Economics, University of Illinois (1970).

THREE STAGE LEAST SQUARES ESTIMATION

I. General Description

The THREE STAGE LEAST SQUARES ESTIMATION program calculates and prints out three stage least squares estimates and an asymptotic covariance matrix. A raw data covariance matrix and two stage least squares residual covariance matrix are the necessary input. Calculations are carried out as in "Econometric Theory" by Arthur S. Goldberger, pp. 347-352. The coefficients may also be stored for use with the ECONOMETRIC REDUCED FORM AND RESIDUAL ANALYSIS program.

References:

Goldberger, Arthur S., Econometric Theory, New York, John Wiley and Sons, Inc. 1964.

Johnson, J., Econometric Methods, New York, McGraw-Hill Book Company, Inc., 1960.

II. Parameters

The parameters appear on the program card following the mnemonic THREE in the following order:

Parameter Number	Description
1	Input Address for raw data covariance matrix. SEQUENTIAL 1-15. Same as output address for K1CLAS.
2	Output Address for coefficients. SEQUENTIAL 1-15.
3	Input Address for residual covariance matrix. SEQUENTIAL 1-15. Same as output address for ECON.
4	Number of equations to be estimated.
5	Number of exogenous variables.

Subparameters

For each equation a card specifying the variables in the equation must follow the main parameter card with the following parameters:

Parameter Number	Description
1	Number of exogenous variables in the equation
2	Number of endogenous variables in the equation.

Parameter Number	Description
3 to N + 2	Variable number of the N variables included in the equations with exogenous variables first; endogenous variables next, with the variable on which the system is normalized last.

III. Special Comments

Although the raw data deck is arranged with exogenous variables first and endogenous variables last, the endogenous coefficients are printed out first, followed by the exogenous coefficients.

The THREE STAGE LEAST SQUARES ESTIMATION program requires input from several other SOUPAC programs. The following is an example of the steps needed to calculate the necessary input.

IV. Example

```
K1CLAS(T1)(T2)(0)(0)()(1)(1).
ENDP
K2CLAS(T2)(T3)(T4)()(8)(2)*1.*.
(4)(2)(1)(2)(3)(4)(1)(2).
(4)(2)(5)(6)(7)(8)(2)(1).
ENDP
ECON(T3)(T2)()(8)()(T5).
THREE(T2)(T6)(T5)(2)(8).
(4)(2)(1)(2)(3)(4)(1)(2).
(4)(2)(5)(6)(7)(8)(2)(1).
ENDP
```

Notice that the equation control cards for both K2CLAS and THREE STAGE LEAST SQUARES must be in the same order.

Also notice that an ENDP card is required after the equation cards.

THE TRANSPORTATION PROBLEM

I. General Description

The "transportation problem" is a special case of linear programming and is of interest because of its computational simplicity. Many economic and business applications of this computation technique have nothing to do with transportation. The name is derived from its original formulation. The essence of the problem can best be described by a simple example.

Suppose a manufacturer has 3 factories and he supplies 5 locations Suppose that the cost per unit from each factory to each location is given. Also assume that the capacity of each factory is given and the amount demanded is equal to the total capacity of the manufacturer. The transportation problem is to find the minimum total cost to ship the capacity of all 3 factories to the 5 demand locations.

The following tabled example should make this clearer (taken from reference [1]).

		Demand Locations					
		1	2	3	4	5	Capacity
Factory	1	$10	$15	$20	$20	$40	50
	2	20	40	15	30	30	100
	3	30	35	40	55	25	150
Amount Demand		25	115	60	30	70	300

The amount demanded row has the amount demanded at each location. The capacity column contains the amount available at each factory. The middle matrix is the cost matrix of shipping one unit of goods from factory i to location j where i = 1, 2, 3, and j = 1, 2, 3, 4, 5. Notice that the sum of the capacities must equal the sum of amount demanded.

The idea is to minimize the total transportation cost while specifying that all goods must be shipped and all demands must be satisfied. The computation technique used to solve the above problem is described in most linear programming texts (see references).

II. Parameters

The parameters follow the letters TRN.

Parameter Number	Description
1	Input address of the supply capacities. CARDS and SEQUENTIAL 1-15.
2	Input address of the amount demanded. CARDS and SEQUENTIAL 1-15.
3	Input address of the cost matrix. CARDS and SEQUENTIAL 1-15.

III. Input

Both the supply capacities and the amount demanded are read in as row vectors. If all the data comes from cards the data decks must be ordered: supply capacities, amount demanded, and then the cost matrix.

IV. Output

Output consists of the optimal transportation order printed in matrix form, the total cost of transportation, and a sensitivity analysis showing the maximum reduction of cost of shipping from supply point i to demand point j and leaving the present solution optimal.

V. Example

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
TRN(C)(C)(C).
END S
DATA (3)(3F5.0)
50.     100.    150.
END#
DATA(5)(5F5.0)
25.     115.    60.     30.     70.
END#
DATA(5)(5F5.0)
10.      15.    20.     20.     40.
20.      40.    15.     30.     30.
30.      35.    40.     55.     25.
END#
/*
```

VI. References

[1] Dorfman, R., Samuelson, P. A., and Solow, R. M., Linear Programming and Economic Analysis.

[2] Hadley, G., Linear Programming.

SPECTRAL ANALYSIS SECTION

AUTOCORRELATIONS

I. Description

AUTOCORRELATIONS is a program designed to perform univariate spectral analysis of time series data. No cross spectral calculations are made by AUTOCORRELATIONS.

II. Input

The input to AUTOCORRELATIONS may be from cards or sequential storage. Each time series must be input as a column vector. If a matrix, each column of which is a time series, is input to AUTOCORRELATIONS, a univariate analysis will be performed on each column.

III. Usage

The parameter string for AUTOCORRELATIONS is

Parameter Number	Description
1	Input address. Cards or Sequential 1-15.
2	Minimum number of lags for which spectral estimates are to be calculated. This value must be ≥ 2.
3	Maximum number of lags for which spectral estimates are to be calculated. This value must be $\leq$ (series length - 2).
4	Incremental value by which the value of parameter 2 steps up to the value of parameter 3.
5	Number of lags, including 0, for which the autocovariances are desired. If this value is less than parameter 3, then it is ignored.
6	An eight column matrix with each row consisting of, from left to right, the frequency, the autocovariance for the corresponding lag, the autocorrelation coefficient for the corresponding lag, the raw spectral estimate, the raw spectral density estimate, the smoothed spectral estimate, the smoothed spectral density estimate, and the log (base 10) of the smoothed spectral density estimate can be output to Sequential 1-15. The final matrix output consists of the vertical augmentation of all the submatrices generated according to parameters 2-4.

IV. Printout

The printout, all of which is by default in AUTOCORRELATIONS, consists of the autocovariances and autocorrelations of the series up to the number of lags necessary to calculate the spectral estimates specified in parameter 3, or up to the number of lags specified in parameter 5 in case it is larger than parameter 3, the mean and variance of the series, and the data referred to under parameter 6 above.

V. Calculations

The autocovariances are estimated by

$$C_k = \frac{1}{N} \sum_{t=1}^{N-k} (X_t - \bar{X})(X_{t+k} - \bar{X})$$

where

$$\bar{X} = \frac{1}{N} \sum_{t=1}^{N} X_t$$

and N is the series length.

The "raw" spectral estimates R(f), f=0, 1/2M, 1/M, 3/2M,..., 1/2 are given by

$$R(f) = 2(1 + 2 \sum_{k=1}^{M-1} C_k \cos 2\pi fk)$$

The smoothed spectral estimates are given by

$$S(f) = 2(1 + 2 \sum_{k=1}^{M-1} C_k w_k \cos 2\pi fk)$$

where the w_k's are the Tukey-Hanning smoothing weights, and

$$w_k = \frac{1}{2} (1 + \cos \pi k/M)$$

The respective densities are obtained by dividing R(f) and S(f) by the sample variance of the series.

VI. Example

Suppose we wish to obtain the spectral estimates for a series of length 400 for from 10 to 30 lags in steps of 4 lags. Furthermore, suppose the series is punched on cards in 20F4.0 format. A possible SOUPAC program would be:

```
/*ID  (accounting information)
//  EXEC  SOUP
//SYSIN  DD  *
MATRIX.
MOVE(C)(S1).
TRA(S1)(S2).
ENDP
AUTO(S2)(10)(30)(4).
ENDS
DATA(400)(20F4.0)
  .
  .
  .     data deck
  .
  .
  .
END#
/*
```

II. References

Bendat, J. S., and A. G. Piersol, Measurement and Analysis of Random Data, John Wiley and Sons, New York, 1966.

Blackman, R. B., and J. W. Tukey, The Measurement of Power Spectra, Dover Publications, New York, 1958.

Fishman, G. S., Spectral Methods in Econometrics, Harvard University Press, Cambridge, Mass., 1969.

Granger, C. W. J., and M. Hatanaka, Spectral Analysis of Economic Time Series, Princeton University Press, Princeton, N.J., 1964.

Jenkins, G. M., and D. G. Watts, Spectral Analysis and Its Applications, Holden-Day, San Francisco, 1969.

Yaglom, A. M., Stationary Random Functions, translated by Richard Silverman, Prentice-Hall, Englewood Cliffs, N.J., 1962.

CROSPA

I. General Description

CROSPA (Cross-Spectral Analysis) is a program made up of a number of spectral analysis subroutines originally written at Princeton and which has been organized and adapted to the SOUPAC system for the purpose of cross-spectral analysis of multivariate time series. The SOUPAC office wishes to thank Dr. R. M. Leuthold and Tom Jarvis for their assistance in obtaining the subroutines comprising CROSPA.

For each time series input to CROSPA, the autocovariances up to a number specified by the user, raw spectral density estimates, and smoothed spectral densities are calculated and printed. A cross-spectral analysis is performed for each of the possible pairs from those time series input to CROSPA. Cross-covariances, raw and smoothed cospectral density estimates, raw and smoothed quadrature spectral density estimates, cross amplitude spectrum density estimates, gain, phase and square coherency estimates are all calculated and printed.

II. Usage

The parameter string for CROSPA is:

Parameter Number	Description
1	Input address. CARDS or SEQUENTIAL 1-15.
2	Input address for filter coefficients (see Section IV, Filtering, below). Blank, Cards, or Sequential 1-15. Usually parameter 2 will be blank.
3	Number of lags to be used in calculations. To perform an analysis with a different number of lags, CROSPA must be called again.

III. Input Array

The time series must be input to CROSPA as columns of a matrix. Each time series in the matrix must be of the same length.

IV. Filtering

In some cases the user will wish to filter the series input to CROSPA. CROSPA constructs a linear filter using the coefficients at the address given by parameter 2 and then performs all the subsequent cross-spectral analyses on the input series after they are transformed by this filter. These coefficients must be stored as a row vector.

As a final step, CROSPA constructs "recolored" univariate spectral estimates for the original time series.

The filtering option should only be used with caution by those users experienced with spectral analysis.

V. Comments

The raw spectral estimates are the unweighted finite Fourier transforms of the auto- and cross- covariances. The smoothed estimates are calculated with Tukey-Hanning weights.

The auto- and cross-covariance estimates are calculated using n-p as the divisor, where n is the number of observations, and p is the number of lags.

VI. Example

Suppose that the matrix of time series resides on S1, that the user wishes to input the filter coefficients 0.5, 0, -0.5 from a data card punched in 3F5.0 format, and that 20 lags are to be used in the calculations. Such a program might be:

```
/*ID     [accounting information]
//  EXEC  SOUP
//FT11F001  DD     [information to define S1]
//SYSIN  DD  *
CROSPA(S1)(C)(20).
ENDS
DATA(3)(3F5.0)
  0.5   0  -0.5
END#
/*
```

VII. References

Bendat, J. S., and A. G. Piersol, Measurement and Analysis of Random Data, John Wiley and Sons, New York, 1966.

Fishman, G. S., Spectral Methods in Econometrics, Harvard University Press, Cambridge, Mass., 1969.

Granger, C. W. J., and M. Hatanaka, Spectral Analysis of Economic Time Series, Princeton University Press, Princeton, New Jersey, 1964.

Jenkins, G. M., and D. Watts, Spectral Analysis and Its Applications, Holden-Day, San Francisco, 1969.

SCALE ANALYSIS PACKAGE

CLIQUE ANALYSIS

I. General Description

This routine is designed to enumerate all third order or higher interrelationships (communication chain) which exist in a sociometric matrix. The algorithm is identical to the method described by Harary and Ross.[1] A communication chain is considered to be any submatrix of order three or more in which all the off diagonal cells are full.

II. Restrictions

The maximum dimensions for an input array are 200 x 200. Input may come from cards or any temporary storage area. The array must contain only zeroes and ones in its elements. Any number greater than zero is considered to be one; therefore, care should be used in constructing the array. Symmetry in the input matrix is not necessary since the program automatically forces symmetry through element-wise products. It is suggested that TRANSFORMATIONS be used to modify input arrays when various cut-off points are used to distinguish ones from zeroes.

III. Parameters

The name CLIQUE ANALYSIS appears first on the program call card and is followed by the following parameter:

Parameter Number	Use or Meaning
1	Input Address of data array. CARDS or SEQUENTIAL 1-15.

IV. Special Comments

The following is an illustration of the clique detection concept.

Data matrix:

0	1	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	1	1	1	0	0	0
0	0	1	0	1	0	1	0	0
0	0	1	1	0	1	1	0	0
0	0	1	0	1	0	1	1	1
0	0	0	1	1	1	0	1	1
0	0	0	0	0	1	1	0	1
0	0	0	0	0	1	1	1	0

Clique (1) 1, 2, 3
Clique (2) 8, 6, 7, 9
Clique (3) 4, 3, 5
Clique (4) 3, 5, 6
Clique (5) 4, 5, 7
Clique (6) 5, 6, 7

[1]Harary and Ross, "A Procedure for Clique Detection Using the Group Matrix", Sociometry, Vol. 20, No. 3, 1956, pp. 205, 215.

PAIRED COMPARISONS

I. General Description

Paired comparisons is a method of obtaining empirical estimates of the form "stimulus j is judged greater than any other stimulus i." Each stimulus in turn serves as the standard; that is, all possible pairs of stimuli are compared. With n stimuli, there are n (n-1)/2 pairs. Comparisons of a stimulus with itself is disregarded; it is assumed that a proportion of 0.50 would result. In the following m = no. of subjects = sample size.

Each subject's preferences are tabulated, and the total number of times he preferred each stimulus is computed producing, A_k, an n x n matrix of 1's and 0's , k = 1, m. Totals for each stimulus and a grand total are computed for each subject and this m x n + 1 matrix is referred to as individual preference sums. The individual tables, A_k, are summed over all subjects to form an n x n frequency matrix F, whose elements (f_{ij}) denote the observed number of times stimulus j was judged greater than stimulus i.

The matrix of proportions, P, is then computed from F, so that p_{ij} is the observed proportion of times stimulus j was judged greater than stimulus i. The matrix X is derived from P by reference to the normal curve; x_{ij} is the unit normal deviate corresponding to the element p_{ij}. These are the sample estimates of the values required to determine the scale values of the stimuli. The scale values are computed by summation producing s_j, a least squares estimate of the scale value of stimulus j.

II. Input

A. Both an indication of ordering for each pair and an array of subjects' choices are required. The former must be given as a set of pair subparameter cards and the latter as an observation of data for each subject in a data deck.

B. In the subjects deck one number is used to denote the subject's choice for each pair. This choice may be "is greater than," "is better than," "is brighter than," etc. This number is 1 if the subject chose the left, or first stimulus, 2 if the subject chose the right or second stimulus. No other coding is acceptable.

C. The pair cards consist of one mention each of every possible pair of stimuli. The order of the pairs is the same as the order of the subjects' choices, i.e. pair 1 corresponds to item 1 of the subject array. The order of the elements in the pairs is reflected in the subjects' choice deck, if (5,7) corresponds to a 1 then the subject chose stimulus 5 over stimulus 7, if (7,5) were the pair, a 2 corresponding would mean stimulus 5 preferred. Note that one set of pair specifications serves for all subjects.

III. Formulas and Calculations

A. INDIVIDUAL PREFERENCE SUMS

Let A be an individual preference frequency table, a_{ij}, is an element of A, i=1, n, j=1, n, where n is the number of stimuli.

$a_{ij} = 1$ if an individual chose stimulus j over stimulus i.

$a_{ij} = 0$ if the individual chose stimulus i over j.

$a_{ii} = a_{jj} = 0$ no stimulus is compared with itself.

$$\text{Individual preference sum for stimulus } j = \sum_i a_{ij}$$

Error messages concerning incorrect frequency tables refer to the configurations of Table A. A can be correct only if subject data and pair cards are correct.

B. STIMULUS PREFERENCE FREQUENCY TABLE, F

Given the matrix A for each of m subjects

$$\text{Stimulus preference frequencies} = f_{ij} = \sum_{k=1}^{m} a_{ij}$$

C. TABLE OF PROPORTIONS, P

If m is the sample size, i.e. number of subjects, then,

$$p_{ij} = \frac{f_{ij}}{m}$$

D. TABLE OF NORMAL DEVIATES, Z

Let p_{ij} be an element of the table of proportions:

Then let

$$e_{ij} = \sqrt{\log\left(1/p_{ij}^{\,2}\right)}$$

$$z = e_{ij} - \frac{2.515517 + .802853xe_{ij} + .010328xe_{ij}^2}{1. + 1.432788xe_{ij} + .189269xe_{ij}^2 + .001308xe_{ij}^3}$$

producing a z for each e_{ij}. Critical values of p occur at 0, 1 and .5 so adjustments are made for these values before the formula is applied and sometimes after.

E. SCALE VALUES, S

$$s_j = \frac{\Sigma z_{ij}}{n}$$, where n is the number of stimuli

Then total scale = $\sum_j s_j$

A row of scales and a total of length n + 1 is calculated.

IV. Output

Matrices for individual preference totals and S, scale values, are always printed. Other intermediate results F, P and Z may be printed on option. All matrices may be stored on option. All results are printed in F format. F, P and Z are n x n matrices, individual totals and S are m x n + 1 and n + 1 respectively (see Section III for calculations). The A matrix is printed only in an error situation.

V. Restrictions

A. The maximum number of stimuli is 42. There is no restriction on the number of subjects. Each stimulus must be paired with each other one. Subjects should have complete data.

B. No more than 300 pairs should be specified per pairs statement. Additional pairs statements may be inserted to a maximum of 881 pairs (42 stimuli).

C. Caution: The number of stimuli, n, and the number of pairs, q, are in the relation

$$q = \frac{n(n-1)}{2} .$$

Any other relationship is invalid.

D. Note if (5,7) is a pair, then (7,5) is invalid. Also, if this is the first pair then the subject's first choice specification concerns stimulus 5 vs stimulus 7; (5,5) is invalid.

VI. Parameters

After the program name, PAIRED COMPARISONS, on the call card come the parameters in the following order:

Parameter Number	Use or Meaning
1	Input Address of data. CARDS, SEQUENTIAL 1-5.
2	Output Address of individual preference sums. Always printed. SEQUENTIAL 1-5.
3	Ω Output Address of stimulus preference frequency table SEQUENTIAL 1-5 and/or PRINT; if not desired, leave parameter blank.

Parameter Number	Use of Meaning
4	Ω Output Address of proportions. SEQUENTIAL 1-5 and/or PRINT; if not desired, leave parameter blank.
5	Ω Output Address of normal deviates. SEQUENTIAL 1-5 and/or PRINT; if not desired, leave parameter blank.
6	Scale Values. SEQUENTIAL 1-5, always printed.

Ω It is possible to punch the output from these parameters while executing this program. If you need this option, see the section in the Introduction on Input and Output. Any storable output may be punched using the Matrix program.

VII. Examples

A.

```
/*ID <accounting information>
// EXEC  SOUP
//SYSIN  DD  *
PAIRED COMPARISONS (C)( )(P)(P)(P).
PAIRS (1,2)(3,1)(4,1)(3,2)(2,4)(4,3).
END P
END S
DATA(6)(6F1.0)
122211
222111
121122
.
.
.
END #
/*
```

Print has been indicated for all output except individual preference sums and scale values which are always printed.

The pairs card indicates that there are 4 stimuli. All possible pairs of these stimuli are presented to the subjects, and the subject's responses are recorded in the order (1,2), (1,3), (1,4), (2,3), (2,4), (3,4). Some of these pair members have been inverted indicating that no special order is required, left member or right member preference of subjects would, of course, be affected by the inversion.

The pairs need not be given in the increasing order of the example, but at all times the order of the pairs is the order of the corresponding subject responses.

The data deck is a set of subject responses for each pair of stimuli.

B.

```
/*ID<accounting information>
//  EXEC  SOUP
//SYSIN  DD  *
TRA (C).
CON(900)*1*.
ADD (1,625)(900)(1,625).
OUT(S1)(1,625).
END P
PAI(S1)( )(S2/P)( )(P).
PAI(5,3)(2,8)(4,16)(7,26)(8,3)(.....)(4,13).
PAI(25,24)(23,28)(.....)(4,12).
END P
.
.
.
.
.
END S
DATA(325)(75F1.0)
     Data Deck--5 cards per subject
END#
/*
```

This example shows a program for 26 stimuli, 26 x 25/2 = 325 is the number of pairs required and the number of subject preferences. Since no more than 300 pairs may be given per pairs statement, at least two pairs statements are needed; two are shown. The unique pairs may occur in any order, the subject responses are in the same order.

The TRANSFORMATIONS program shown is designed to correct subject responses punched zero/one or blank/one to 1 and 2.

A selection of possible output has been made. Note that individual preference sums and scale values, as well as normal deviates and stimulus preference frequencies are printed. The latter is also stored. This storage implies some further use is made of the frequencies, perhaps in the missing part of the program.

VIII. References

Torgerson, Warren S. Theory and Methods of Scaling. John Wiley and Sons, New York; 1960, pp. 166-173.

Edwards, Allen L. Techniques of Attitude Scale Construction. Appleton Century, Crofts, New York: 1957, pp. 19-52.

SCALOGRAM ANALYSIS

I. GENERAL DESCRIPTION

The SCALOGRAM ANALYSIS (mnemonic: SCA) was developed to provide a method of producing Guttman scales automatically without the need of external decisions to determine which items do and which items do not enter into Guttman scales. Items are grouped together in as few as possible submatrices with each subgroup having a maximum homogeneity within each submatrix. Each item from the total group is chosen to fit into only one submatrix.

The SCALOGRAM program is started by choosing an item from the total group and then searches the remainder of the items to find an item similar to the item chosen. Similarity is tested by using an error criteria and a chi-square test to insure that the items are similar. If the above criteria are met, this item is added to the first item and a scale is formed. This last item is then used to find another similar item and this procedure continues until either of the two criteria is not met. Whenever a criteria fails, the scale is terminated and a new scale is started.

SCALOGRAM will only work for dichotomous data and it can be used to analyze both subject-wise and item-wise. SCALOGRAM differs from Guttman analysis in three ways: 1) It uses an empirical rather than a rational basis for selecting items to enter a scale; 2) It uses a statistical method of deciding on groups and for testing the scale-ability of the item; 3) It yields multiple scales rather than reject the scale hypothesis for the whole item set.

SCALOGRAM can be considered to be more descriptive than the raw data but less than factor analysis. SCALOGRAM also is unlike factor analysis in that SCALOGRAM is not bound to linear assumptions about the regressions involved. Factor analysis is set up to study quantitative variables and will not show correct relationships between qualitative variables, SCALOGRAM will show what relationships do exist between qualitative variables. (See Guttman 1950 for a complete discussion of the relation between the scalogram technique and other statistical procedures.) (See Lingoes 1963 for the complete algorithm for SCALOGRAM.)

II. REFERENCES

Guttman, L. "Relation of Scalogram Analysis to Other Techniques." In Samuel A. Stouffer, et al., Measurement and Prediction. Princeton, N.J.: Princeton University Press, 1950 (pp. 172-212).

Lingoes, J.C. "Multiple Scalogram Analysis. A Set-Theoretic Model For Analyzing Dichotomous Items." Educational and Psychological Meausrement XXIII (1963), 501-524.

Lingoes, J. C. "A Multiple Scalogram Analysis of Selected Issues of the 83rd U.S. Senate." American Psychologist, XVII (1962), 327.

III. INPUT

Input to the SCALOGRAM ANALYSIS Program consists of a rectangular array of dichotomous variables. Scaling is done on columns. Thus an N x M array of data will be scaled across the N "subjects" and yield up to M scales. To scale on the M "items" set the transpose flag in SCALOGRAM (see parameters).

Zero and one are the usual values of the dichotomy. Two is taken as missing data and distributed randomly among the other codes. Blanks are zeros. In general 2 is missing data, 1 is one level of the dichotomy and "anything else" is the other level of the dichotomy.

Labels may consist of up to 28 characters, one label per card. If both labels and data are read by SCALOGRAM from cards, the labels deck goes first. If SCALOGRAM is to scale subjects, labels must refer to subjects. Note if labels are not used considerably more core is available for variables and subjects. If labels are used the data card should be: DATA(n)(nA4) where $n \leq 7$.

IV. PARAMETERS

The program mnemonic is SCA. The following parameters appear on the program card:

Parameter	Description
1	Input Address
2	Address of Labels
3	A 1 indicates that the matrix should be transposed

Since the program scales by columns or items, to scale by subjects, indicate in parameter 3.

V. RESTRICTIONS

Data must be dichotomous (see input section). If data is not of this form, TRANSFORMATIONS may be used to recode it.

VI. EXAMPLES

```
SCA(C)(C)(1).
ENDS
DATA(7)(7A4)
  :
  :    (labels)
  :
END#
DATA(40)(40F1.0)
  :    (data)
  :
END#
```

Labels and data are on cards, 28 columns are used for labels and scaling will be done by rows.

```
SCA(S1).
ENDS
DATA(30)(30F1.0)
  .
  .
  .    (data)
  .
  .
  .
END#
```

Data is on SEQUENTIAL 1 and scaling will be done by columns.

PROBIT ANALYSIS SECTION

PROBIT (Mnemonic: PRB)

I. General Description

This program calculates maximum likelihood estimates for the parameters A and B in the probit equation:

$$Y \quad A + BX$$

An iterative scheme is used.

II. Restrictions

The input vectors must be equal length k and: $3 \leq k \leq 3000$. Each input vector comes from a separate input address.

III. Parameters

Parameter Number	Use or Meaning
1	Input vector of dosage level. CARDS or SEQUENTIAL 1-15.
2	Input vector of number of subjects tested at each dose level. CARDS or SEQUENTIAL 1-15.
3	Input vector containing the number of subjects at each level responding to the drug. CARDS or SEQUENTIAL 1-15.
4	Output vector of length k containing the proportion of subjects responding to the various close levels of the drug. SEQUENTIAL 1-15, and/or PRINT.
5	Output vector of length k containing the values of the expected probit for the various levels of the drug. SEQUENTIAL 1-15 and/or PRINT.

Printed output consists of:

1 - Estimate of intercept constant A

2 - Estimate of probit regression coefficient B

3 - Chi-square value for a test of significance of final probit equation

$$X^2 = \sum_{i=1} \frac{(R_i - N_i P_i)^2}{N_i P_i (1 - P_i)}$$

where R_i = number of responses (input address 3)
N_i = number of objects tested (input address 2)
P_i = cumulative normal distribution values corresponding to Z_i where $Z_i = (A + BX_i) - 5$
where A and B are from final probit equation

4 - Degrees of freedom for X^2
d.f. = k - 2

References:

D. J. Finney, Probit Analysis, Second Edition, (Cambridge University Press 1952).

The program was adapted from the IBM Scientific Subroutine Package, 360A-CM-03X, Version III, page 44.

IV. Example

If two or more input addresses are cards, the cards must be stacked in order of their parameter numbers. For example:

```
/*ID
//  EXEC  SOUPAC
//SOUPAC.SYSIN  DD  *
MAT.
MOVE(CARDS)(SEQ2)
END P
PRB(CARDS)(SEQ2)(CARDS)(PRINT).
END S
DATA(1)(.....)
  .
  .     cards for SEQ 2
  .
END#
DATA(1)(.....)
  .
  .     Cards for Parameter 1
  .
END#
DATA(1)(.....)
  .
  .     Cards for Parameter 3
  .
  .
END#
/*
```

NOTE The mnemonic for PROBIT is PRB, nor PRO.

RANDOM NUMBER GENERATION SECTION

RANDOM NUMBER GENERATOR

I. General Description

This program generates a matrix of random numbers or digits from a specified probability distribution.

II. Main Parameter Card

Parameter Number	Description
1	Input Address of 9 (nine) digit integer, used as a starting point for the random number generator. CARDS or SEQUENTIAL 1-15.
2	Output Address of random numbers matrix. SEQUENTIAL 1-15. PRINT is not valid.
3	Number of rows in output matrix of random numbers.
4	Number of columns in output matrix of random numbers.
5	Output Address of 9 digit integer which is finishing point of the random number generator. Do not specify PRINT since number is automatically printed.

III. Subparameter Cards

To specify the distribution of the random numbers, choose one of the subparameter cards listed below. Refer to Appendix E for definition of these distributions, as well as information on obtaining distributions not given below.

BINOMIAL (N)*P*.	The sum of N independent trials, each with probability P of success.
RECTANGULAR *θ1**θ2*.	Sometimes called the continuous uniform distribution. The probability of each interval in [θ1,θ2] of fixed size is the same.
NORMAL *N**σ^2*.	Here N is the desired mean of the normally distributed variables, and σ^2 the variance.
GAMMA *α**β*.	The Gamma distribution with parameter β and α degrees of freedom. α should be integer valued. Note that Gamma * $\frac{n}{2}$ ** β*. is the chi-square distribution with n degrees of freedom, while Gamma *1.**λ*. is sometimes called the negative exponential distribution with parameter λ.
DISCRETE(N).	Yields random digits from 0 to N, each with equal probability of occuring.

IV. Special Comments

If this program is used with the same integer starting point, it will generate the same numbers. Thus, use Parameter 5 to output the finishing location, and then pass that address as the starting location for the next use of this program.

It should be noted that time requirements for generating random numbers will vary greatly among the various distributions. More specific information is available from the SOUPAC Office.

V. References

IBM System/360 Scientific Subroutine Package (360A-CM-03X) Version 2, page 54.

RR

UTILITY

UTILITY PROGRAM

I. General Description

The UTILITY program has been designed to handle small utility functions which do not necessitate or justify the creation of a unique program within the SOUPAC system. The following statements will invoke the UTILITY program.

```
UTILITY.
(insert subparameter card or cards here
END P
```

The following sections describe the functions of the various subparameters.

II. PRESORT Program

The PRESORT Program is presently the only program in the UTILITY program. It is used to set up the data cards to be input into the IBM SORT/MERGE package which will be executed following the present SOUPAC program and before another SOUPAC program which will use the sorted data for an input.

SORT (0 or 1)(0 or 1)(V_1).....(V_n).

Parameter Number	Use or Meaning
1	0 if data is to be sorted in ascending order. 1 if data is to be sorted in descending order.
2	0 if data to be sorted is in single precision. 1 if data to be sorted is in double precision.
3 through n $\leq$ 20	indicates the variable or variables to be sorted with the later variables, if any, varying most rapidly.

Following the SOUPAC program in which the UTILITY program appears, the following card must appear:

```
// EXEC SOUPSORT,INPUT=Snn,OUTPUT= Smm.
```

where nn and mm represent the two digit equivalent of the sequential unit numbers to be input to the sort and output from the sort to the next SOUPAC program. The two units must not be the same.

The next card will start the next SOUPAC program which will operate under the assumption that the sorted data has been supplied on the specified sequential unit in the output of the SOUPSORT program.

```
// EXEC SOUPAC,DISP=OLD
//SYSIN  DD  *
.
.  (Your program which uses the sorted data).
.
```

The example given below is for sorting cards input data so that it may be input into a FREQUENCY program which uses variable 5 as a control variable.

```
/*ID identification card information
//  EXEC  SØUPAC
//SYSIN  DD  *
MATRIX.
MØVE(CARDS)(S1).
END PRØGRAM
UTILITY.
SØRT(0)(1)(5).
END PRØGRAM
END SØUPAC
DATA(10)(10F5.0).
.
.     (user's data deck)
.
.
END #
/*
//  EXEC  SØUPSØRT,INPUT=S01,ØUTPUT=S02
//  EXEC  SØUPAC,DISP=ØLD
//SYSIN  DD  *
FREQUENCY(S2).
TWØ.
PER(1)(1)(1).
CØNTRØL(5).
END PRØGRAM
END SØUPAC
/*
```

The data on S1 is sorted in ascending order on variable 5. The data is passed to the SOUPSORT job step on S1 in double precision. This data is sorted on variable 5 and then output onto S2 in double precision. It is then input into the FREQUENCY program of the next SOUPAC job step, whereupon analysis continues.

III. Notes, Restrictions, and Ideas

1. The default output from MATRIX is in double precision
2. The output from TRANSFORMATIONS is in single precision
3. Only one utility program is allowed per SOUPAC program
4. If other sequential units have been used during the first SOUPAC program besides the one passed to the sort job step, they are still intact and usuable in the second SOUPAC program due to the DISP=OLD parameters.

APPENDIX A

OPTIONS TO A SOUPAC JOB:
PARMS, PROLOG CARDS, AND $-CONTROL CARDS

A. PARMS

Parms are arguments to the keyword 'Parameter,' contracted into the keyword 'PARM', which give instructions to a processor running under a 360-system. In this context, SOUPAC is a processor running under a 360 system. PARMS are always coded on an EXEC card and have the following form:

```
// EXEC SOUPAC,PARM='OPT1,OPT2,....,OPTm'
```

The permissable options to be used as SOUPAC PARMS are listed below with an explanation of their use and function. Note that the default is underlined, that is, // EXEC SOUPAC is equivalent to // EXEC SOUPAC,PARM='OPT1,OPT2...' where the underlined PARM is to be taken as one of the list of options in the PARM string in the example. These PARMS give the SOUPAC system instructions in the same way that parameters give SOUPAC statistical or data management programs instructions.

1. NODYNAM or DYNAM

   NODYNAM implies that a non-dynamically allocatable version of the library of statistical procedures is to be used. This version will run in some 150K of core and will handle a lesser number of variables than the dynamically allocatable version. DYNAM will use the dynamically allocatable version of any program requested which will handle more variables in an arbitrarily specified amount of core above a certain minimum. If using DYNAM, see the SOUPAC consultants for a handout on optimal region sizes for particular numbers of variables.

2. EXECUTE or NOEXECUTE

   NOEXECUTE implies that the SOUPAC parameter deck, for which the Syntax Interpreter is to scan and build intermediate parameters, should not be executed. NOEXECUTE indicates that only a syntax check is to be performed. If EXECUTE is specified and no errors are found by the Syntax Interpreter, the job step will proceed. If EXECUTE is specified and errors are found by the Syntax Interpreter, execution of the step may continue depending upon whether LET or NOLET is also specified.

3. NOLET or LET

   If an error is found by the Syntax Interpreter and EXECUTE has been specified, execution will proceed only if LET was also specified. In this case, execution will proceed only through the last program processed which was completely error free. If NOLET was specified and errors are found by the Syntax Interpreter, execution will not be permitted.

4. LIST or NOLIST

   LIST indicates that all program cards are to be listed. NOLIST indicates that only the prolog section of the SOUPAC parameter deck is to be listed.

5. PGM or NOPGM

PGM indicates that a complete SOUPAC parameter deck and data decks follow. NOPGM indicates that only the prolog section and data deck follow, and that the intermediate parameters are being provided by the user by over-riding the cataloged procedure. This implies that the user has previously run a SOUPAC job and has saved the two necessary data sets so that he may run the same program again. To perform this saving of data sets correctly, a user should visit the SOUPAC office first to ensure it is done correctly.

If any error is found by the Syntax Interpreter in the prolog section, the job step will not continue.

If the job step which generated the intermediate parameter data sets found syntax errors, execution of the job step in which NOPGM is specified will continue (If EXECUTE is specified) through the last program processed which was completely error free regardless of whether LET or NOLET was specified in either job step.

Examples:

To do just a syntax check:

```
//  EXEC  SOUPAC,PARM='NOEXECUTE'
```

To execute up to the first program found to have syntax errors:

```
//  EXEC  SOUPAC,PARM='LET'
```

To execute up to the first program found to have syntax errors and use the dynamically allocatable library:

```
//  EXEC  SOUPAC,PARM='LET,DYNAM'.
```

Note that the PARMS may be listed in any order.

B. PROLOG OF A SOUPAC JOB

Described below are several # control cards which may appear in the prolog of a SOUPAC job. Within the prolog these control cards may appear in any order. If prolog control cards are used, they must appear immediately after the SYSIN card. The Syntax Interpreter determines the end of the prolog when it reads a card which is not one of these types. All types have parameters and must be terminated by a period. Prolog cards may not have continuation cards, hence all parameter information must be punched within 80 columns. There is no limit to the number of prolog cards permitted nor is there any restriction on the number of any one type. If conflicting information is entered, the information entered last overrides any previous definitions.

1. #REPEAT OPTION

The #REPEAT OPTION is used to repeat sections of a SOUPAC parameter deck an optional number of times. The #REPEAT card

which appears in the prolog section will be followed by up to 22 (twenty-two) integer parameters which will indicate the number of repetitions of up to 22 repeat sequences. The card sequences to be repeated will be preceded and followed by #SREP and #EREP cards respectively. Example:

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
#REPEAT (2).
<additional program cards>
#SREP
CORRELATION (C)( )(S1).
SQUARE ROOT FACTOR ANALYSIS (S1)(P(F))(20)(C)(P(F)).
#EREP
END S
```

In this example the program sequence of CORRELATION and SQUARE ROOT FACTOR ANALYSIS will be repeated twice. Four card input data sets would be required for the repeated sections.

Repeat sequences which begin before a main program and end in a subprogram or which begin in a subprogram and do not end in the same subprogram are not allowed. Nested or overlapping repeat sequences are not allowed. Also a #SREP card cannot be immediately followed by a #EREP card and a single appearance in the deck of either card will cause an error.

2. #V-UNIT OPTION

#V-UNIT allows the user to change input and output addresses in the execution of one SOUPAC job. The form of a #V is as follows:

$$\#Vn\ (m)\ (A_1).\ .\ .\ .\ .\ .\ .\ .(A_k).$$

where n is an integer 1 through 9, thus there can be at most 9 variable addresses, namely V1 through V9; and m is a counter which determines how many times a variable address may be used before it assumes the next value in its list of possible values. $A_1 \ldots\ldots A_k$ are addresses which Vn assumes. These can be any valid address. At the moment, however, forms like (S1/P) will not work. Note that CARDS and PRINT are permitted.

Finally, the list of addresses is cyclic; that is, if, after A_k has been used, Vn occurs again in the program, Vn will have the value A_1, and so on.

```
/*ID
//  EXEC SOUP
//SYSIN  DD  *
#V9(1)(S1)(S2)(S3)(S4).
#V5(1)(S1)(S2)(S3)(S4).
#REPEAT (4).
MAT.
```

(Example, Continued)

```
#SREP
MOV (C)(V9).
#EREP
HOR (V5)(V5)(V5)(V5)(S5).
END P
 .
 .
 .
END S
```

This program segment reads 4 separate card decks, saving them in temporary storage, and horizontally augments them into one data set.

The equivalent without the use of #REPEAT and #V would be as follows:

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
MAT.
MOV(C)(S1).
MOV(C)(S2).
MOV(C)(S3).
MOV(C)(S4).
HOR(S1)(S2)(S3)(S4)(S5).
END P
  .
  .
  .
END S
```

Note that the MOV(C)(V9). statement is expanded into four move statements and V9 takes the values S1 through S4. Similarly, V5 takes on the values of S1 through S4.

3. #OLD OPTION

The #OLD option is used to define the number of rows in a sequential data set created by a previously run SOUPAC job. The number of rows is then entered into a table in the monitor. This option should be used whenever the header record on the data set is not known to have a correct value for the number of rows, and the user does not want to execute a MATRIX MOVE to count the rows. To use the option, punch a card with #OLD in the first four columns. Then code the address and the number of rows in the usual SOUPAC fashion. The number of columns may be coded on the card if desired, but will be totally ignored. Include this card in the prolog section of the SOUPAC job.

For example, to indicate that a data set to be input from SEQUENTIAL 1 has 77 rows you would prepare the following card:

```
#OLD (S1)(77).
```

4. #TEST OPTION

There is also available a #TEST option; however, this facility is complicated and intended for testing purposes within the SOUPAC office and has no significant advantage for the general user.

5. #DEFINE OPTION

Whenever the user wishes to specify the dimensions of a direct access data set (DISK address), punch #DEFINE in the first seven columns of a card followed by the address, number of rows and number of columns coded in the usual SOUPAC fashion. Include this card in the prolog section of your program. For double precision matrices, code the same number of rows, but twice as many columns as otherwise. DISK 1 and DISK 2 have default definitions of 450 rows by 450 columns single precision. If the user desires any other dimensions on these data sets, #DEFINE must be used. If the user desires to use any DISK address other than DISK 1 and DISK 2, #DEFINE must be used besides supplying the necessary DD cards.

For example, to define a data set for DISK 17 with 20 rows and 40 columns double precision, you would prepare the following card:

#DEFINE (DISK 17)(20)(80).

Notice that all prolog cards start with a # in column one and must occur before any SOUPAC program parameter cards. A #-card in the middle of the SOUPAC program parameter deck is treated as a comment. There is, however, a #-control card, while not strictly a prolog card, which may occur in the SOUPAC program parameter deck and will not be treated as a comment. This is the #-zero card and is the only exception to the statement about # cards being comments if in the middle of the deck. The #-zero card is essentially a debugging tool to facilitate reading of dumps if one is needed. It has no particular use for the user.

C. $-CONTROL CARDS

$-CONTROL CARDS are used to provide additional information to a SOUPAC program above and beyond what is included in the parameters. There are 3 $-control cards. All must begin in column one with the character $ and then continue accross the card without blank columns.

1. $C-B

The $C-B card provides as its arguments the variables to be used as control breaks for a program which accept control breaks. The use of this card with a program which does not accept control breaks is an error. The form of this card is as follows:
$C-B($V_1$)($V_2$).....($V_n$).
When V_1 through V_n are variable numbers and n must be less than or equal to 24.

2. $INP

$INP has as its arguments a string of input addresses. The form is:
$INP($A_1$).....($A_n$).
where A_1 through A_n are input addresses including cards. The number of addresses will be determined by the program accepting the $INP card and will explicitly mentioned in the program write-up.

3. $OUT
$OUT($A_1$).....($A_n$).

$OUT has as its arguments a string of output addresses. The form is the same as that for $INP and the number of addresses is also determined by the program accepting the $OUT card. Multiple output address will be accepted. See section on Input/Output multiple addresses.

APPENDIX B

SOUPAC INPUT-OUTPUT AND TEMPORARY STORAGE

I. GENERAL

A. Input and Output as Data Types

Consider a set of data which a researcher wants intercorrelated. To do correlations there is in the SOUPAC library of statistical precedures a correlation program. Input to the correlation program is the researcher's raw data; output from the correlation program is a matrix of correlation coefficients. Similarly, every conceivable program has a particular input; in fact, perhaps several inputs, and some output.

The nature of the input and output of a particular program will depend on the program and its intent. For example, raw data variables are input into a correlation program which outputs a correlation matrix. But a factor analysis program expects as input a correlation matrix, and yields as output a factor matrix. In contrast to the singular relation of the nature of input and output to a particular statistical program, every program finds its input somewhere and must put its output somewhere.

B. Input and Output as Data Sources

SOUPAC is designed in such a manner that the researcher can tell any program where his inputs are and where to put his outputs. Punched cards are an obvious input source; printed pages are an obvious output source. But the nature of a punched card deck input into a correlation program would be that of raw data variables. In the SOUPAC system input and output sources are also called addresses. Thus, a possible input address for a correlation program is cards and a possible output address for correlation coefficients is print. Input and output addresses are parameters to every program in the SOUPAC system. As the researcher reads a particular program write-up he will notice that the order of the parameters determines the nature of his input or output and his supplying an input or output address determines whether or not he uses or gets the particular inputs and outputs.

II. ELEMENTARY INPUT/OUTPUT ADDRESS AND TEMPORARY STORAGE

A. Possible elementary input and output addresses in the SOUPAC system are these:

INPUT: CARDS, SEQUENTIAL 1, SEQUENTIAL 2, . . .
. . . SEQUENTIAL 15

OUTPUT: PRINT, SEQUENTIAL 1, SEQUENTIAL 2, . . .
. .SEQUENTIAL 15 (See section on punched cards).

Again, CARDS and PRINT are obvious sources. SEQUENTIAL 1 through SEQUENTIAL 15, however, are input or output names of 15 temporary storage regions available to the researcher in the SOUPAC system. These 15 temporary storage regions are provided for exactly that purpose, temporary storage of data. Notice that with this facility a user can save his

correlation matrix, for example, at SEQUENTIAL 1 and then give SEQUENTIAL 1 as an input address to a factor analysis program. Or a researcher can construct a copy of his data on temporary storage and then let any number of programs use the same data as input from the same input address, saving him the effort of making multiple copies of his card deck so that each program would read its own deck. Finally, temporary storage addresses enable the saving of intermediate results for further processing or modification by other programs and thereby enable the researcher to construct his own analysis procedure by providing the appropriate inputs and outputs to the right programs at the right times.

B. SOUP vs SOUPAC with Respect to Temporary Storage

There are two ways of invoking the SOUPAC system. One can ask for SOUPAC or SOUP. Note that all 15 temporary storage regions are allocated to SOUPAC, while only SEQUENTIAL 1 through SEQUENTIAL 5 are allocated to SOUP. Asking for SEQUENTIAL 6 through SEQUENTIAL 15 when running under SOUP will cause an error and terminate the job.

All of these input-output addresses may be abbreviated as follows:

CARDS	C
PRINT	P
SEQUENTIAL 1	S1 (or T1)
.	
.	
.	
SEQUENTIAL 15	S15 (or T15)

T1 through T15 are alternative abbreviations for SEQUENTIAL 1 through SEQUENTIAL 15. T1 through T15 are, in fact, abbreviations of TAPE 1 through TAPE 15. SEQUENTIAL 1 through SEQUENTIAL 15 and their abbreviations are the recommended uses. The T1 through T15 notation reflects a real technical distinction but has been kept to enable programs using that notation to run.

C. Multiple Output Addresses

A researcher may want to output to several sources: he may desire to both print and save some results for later use. He cannot, however, input from more than one source for a particular input address. The facility of multiple output addresses has the following construction:

(output address^{1}/output address^{2}/output address^{3}).

This is the completely general form providing for up to three separate outputs. Each output must be a different source, however. Thus, (S1/P/X) is a valid multiple output address providing for temporary storage at S1,

a print of the same data, and a punched copy of the data. (See section on punch for explanation of X). (S1/P) will print and store but not punch. The order of the addresses makes no difference. (S1/P) is equivalent to (P/S1). Forms such as (S1/S2), however, are not permitted, nor are (P/P) or (X/X): one can output only to one sequential and only once to P or X.

The above general form is available only if the output address in the particular program is marked with an Ω.

In all cases, however, the form

(output address 1/output address 2)

is valid unless the program write-up explicitly has a restriction.

D. Print is F Form of Output Print Address.

There is yet another form to output addresses. <u>This form is available only where the researcher finds the symbol Ω in the program write-up</u> and has to do with the kind of printed output. For technical reasons, most programs print in a form called E-format which is a form of scientific notation. This form allows the computer to print numbers of any size. Some programs, for which the output numbers are known to be constrained, as in correlation coefficients, however, print in a form called F-format which is ordinary decimal number representation. F-format generally cannot print numbers larger than a pre-determined size. The size of number depends on the nature of a researcher's data, but the program has no way of knowing this, hence, the most general form, E-format is used.

The researcher however, can on option specify F-format. To print in F-format he would use the following output address:

(P(F)) or (P(F)/S1) if he wanted a multiple

output address. Those programs which print in F-format already, as for correlation coefficients, can be made to print in E-format by using the following output address:

(P(E)) or (P(E)/S15).

The different forms look like this:

E-format	Scientific Notation	F-format	Decimal Number Representation
± 0.12345E 06	± 1.2345×10^5	± 123456.12345	±123456.12345

All four numbers have the same value correct to 5 places. Notice that F-format cannot represent a number greater than 999999.99999 in absolute value whereas E-format can represent the first 5 digits of any number of order of magnitude up to 10^{99} . The numbers of digits illustrated for E and F formats are the pre-determined limits for the size of numbers. E-format is the more general form but F-format is easier to read.

In this example of E-format the E 02 part is to be understood as 10^2. E 03 would be 10^3 and E-04 would be 10^{-4}. Thus, .376 E 03 is .376 x 10^3 or 376. while .1294E-01 is .1294 x 10^{-1} or .01294. The sign following the E determines which way to move the decimal point; left for negative, right for blank or positive. The number following the sign or blank determines how many places to move the decimal point.

E. Punched Output (Don't forget to specify CARDS= on ID Card!)

All programs which have output addresses marked with the symbol Ω can punch output directly by using the X output address. X is the abbreviation for cards as output. C used as an abbreviation for an output address will be an error. Punched output generated by the use of the X output address will be in E-format. (See section above). X(F) is not a valid form and will be an error.

If punched output is desired in a form other than E-format or from a program which does not allow the X output address, then the researcher must make a copy of his data on temporary storage and go to the MATRIX program and use the PUNCH instruction provided in that program.

F. Obtaining Additional Input/Output Sources

It happens that 15 temporary storage locations may not be enough. Additional temporary storage may be obtained by calling for S16 through S40. Use of S16 through S40 requires the addition of Job Control Cards to the 360 system cards of the SOUPAC program deck. At least the first time the researcher should check with SOUPAC consultants before doing this; firstly to learn to do it correctly if he doesn't know how already, and secondly, if he knows how, to make sure none of the Job Control Language has been changed or modified, which can happen due to 360 system changes or reconfigurations, or SOUPAC system changes, which may not be announced, in contrast to SOUPAC program changes which would have been announced.

If in special instances even 40 temporary storage regions are not sufficient or a situation arises where so-called DISK temporary storage is required, there can be made available temporary storage regions called DISK1 through DISK40. Check with the SOUPAC consultants before using these for the proper Job Control Cards and the proper SOUPAC prolog cards.

G. Using Owner Data Sources or Special Input/Output Requirements in the SOUPAC system

Users' own tapes or disk packs can be used with the SOUPAC system for input or output.

Special input/output requirements can usually be handled in the SOUPAC system provided the requirements can be handled by the 360 system at all. In such cases check with the SOUPAC consultants.

General problem types of the nature alluded to above would be multiple file volumes, blocked input/output, formatted or unformatted input/output, different kinds of record lengths and different forms of data representation due to machine differences or differences in facilities at other computer installations.

APPENDIX C

SOUPAC Glossary of Terms on Data Representation

BIT- a binary digit; e.g. a 0 or 1. a BIT has two states.

BYTE- (also called a CHARACTER)--8 bits. A BYTE (CHARACTER) has 2^8 (256) states.

CHARACTER- (see BYTE).

CONVERSION- the process of going from one of the three DATA TYPES to another.
For example, the number 6.25 would be represented
in CHARACTER mode as:
11110000 01001011 11110010 11110101

in FLOATING POINT mode as:
01000001 01100100 00000000 00000000

and in FIXED POINT mode as:
00000000 00000000 00000000 00000110

Notice that in the FIXED POINT representation, the fractional part has been lost.

DATA TYPE- method or mode of representing information. There are three essential DATA TYPES.

1. CHARACTER mode
2. FLOATING POINT mode
3. FIXED POINT mode

Notice that a single bit pattern has different meanings when interpreted under each data type.

For example, the SINGLE WORD
11010111 11000001 11100100 11010011

in CHARACTER mode means:
PAUL

in FLOATING POINT mode means:
$-.3750452400107475 * 10^{28}$
$-3750452400107475. * 10^{12}$

and in FIXED POINT mode means:
-675158829

The radix point is essential to FLOATING POINT representation and does not exist in FIXED POINT representation.

DOUBLE PRECISION- a data attribute similar to LENGTH ATTRIBUTE which specifically indicates that the data item is a DOUBLE WORD in length.
A DOUBLE PRECISION FLOATING NUMBER has 64 bits

a sign bit
a 7 bit exponent
a 56 bit magnitude $\simeq$ 16.8 decimal digits

DOUBLE WORD- two WORDS; also 64 BITS; also 8 BYTES (see WORD). A DOUBLE WORD has 2^{64} states.

FIXED POINT NUMBER- (also INTEGER) a 32 bit data item which takes on only integer values from

(-2^{31}) to $(2^{31} - 1)$

or equivalently

-2147483648 to 2147483647

The range of FIXED POINT NUMBERS can be represented by

FLOATING POINT NUMBER- a number which is to be internally represented in a manner similar to socalled "scientific notation." FLOATING POINT NUMBERS are represented as

$S.M * 16^E$

where

S is the sign + or -.
. is the base 16 radix point
M is the magnitude, where $0 \leq M < 1$.
In SINGLE PRECISION, M is 24 bits long.
In DOUBLE PRECISION, M is 56 bits long.
* is the symbol for ordinary multiplication.
E is a 7 bit exponent.

The range of FLOATING POINT NUMBERS available for both SINGLE PRECISION and DOUBLE PRECISION can be approximately represented by.

Notice that the precision does not affect the overall range of values available. The precision only indicates the number of values which can be represented <u>exactly</u> within the ranges given.

INTEGER- (see FIXED POINT NUMBER)

LENGTH ATTRIBUTE- the number of BYTES in a data element. For example, a DOUBLE WORD has a LENGTH ATTRIBUTE of 8. A BYTE has a LENGTH ATTRIBUTE of 1.

REAL NUMBER- (see FLOATING POINT NUMBER)

SINGLE PRECISION- a data attribute similar to LENGTH ATTRIBUTE which specifically indicates that the data item is one WORD long. A SINGLE PRECISION FLOATING POINT NUMBER has 32 bits.

- a sign bit
- a 7 bit exponent
- a 24 bit magnitude $\underset{\sim}{\sim}$ 7.2 decimal digits.

SINGLE WORD- (see WORD)

WORD- (also SINGLE WORD)--32 bits; also 4 bytes. A WORD has 2^{32} (4294967296) states.

Example: 01000001 00010000 00000000 00000000

APPENDIX D

Densities of Some Common Probability Distributions

A. Definitions

f(x) will be used, in what follows, to denote the probability density function (p.d.f.)

$N(\mu,\sigma^2)$ denotes the normal distribution having mean μ, and variance σ^2. The p.d.f. is given by:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \; e^{\frac{-(x-\mu)^2}{2\sigma^2}} \qquad \text{for } -\infty < x < \infty$$

here we may have $-\infty < \mu < \infty$ and $0 < \sigma^2 < \infty$

$\chi^2(n)$ denotes the chi-square distribution having n degrees of freedom. The p.d.f. is given by:

$$f(x) = \frac{1}{\Gamma(n/2)2^{n/2}} \; x^{n/2 - 1} \, e^{-x/2} \qquad 0 < x < \infty$$

$$= 0 \qquad \text{otherwise}$$

here n is a positive integer, and Γ denotes the well known "gamma function," which is defined by

$$\Gamma(\alpha) = \int_0^\infty y^{\alpha-1} e^{-y} dy \qquad \alpha > 0.$$

$\Gamma(\alpha) = \alpha!$ when α is integer valued (i.e. a positive integer), and we usually define $\Gamma(0) = 1$.

$\gamma(\alpha,\lambda)$ or $G(\alpha,\lambda)$ denotes the gamma distribution, with parameters α (degrees of freedom), and λ. The gamma p.d.f. is given by:

$$f(x) = \frac{\lambda}{\Gamma(\alpha)} \; (\lambda x)^{\alpha-1} e^{-\lambda x} \qquad 0 < x < \infty$$

$$= 0 \qquad \text{otherwise.}$$

Here we require $\lambda > 0$, and $\alpha > 0$. Note that with $\lambda = 1/2$, and integral α, we have $\chi^2(2\alpha)$, while $\Gamma(1,\lambda)$ is usually called the "negative exponential distribution" with parameter λ.

t(n) denotes the t-distribution with n degrees of freedom. It's p.d.f. is:

$$f(x) = \frac{\Gamma(n/2) + 1/2)}{\Gamma(n/2)\sqrt{\Pi n(1+x^2/n)^{n+1}}} \qquad \infty < x < \infty,$$

where n is a positive integer.

$F(n_1, n_2)$ denotes the F-distribution with n_1 degrees of freedom "in the numerator" and n_2 degrees of freedom "in the denominator." The p.d.f. is:

$$f(x) = \frac{\Gamma(n_1/2 + n_2/2)(n_1/n_2)^{n_1/2}\, x^{n_1/2 - 1}}{\Gamma(n_1/2)\,\Gamma(n_2/2)\,(1 + n_1 x/n_2)^{(n_1+n_2)/2}} \qquad 0 < x < \infty$$

$$= 0 \qquad \text{otherwise}$$

where n_1, n_2 are positive integers. The F-distribution arises in practice from the quotient $(x_1/n_1)/(x_2/n_2)$, where x_1 is $x^2(n_1)$ and x_2 is $x^2(n_2)$, hence the terminology "degrees of freedom in the numerator."

Beta(p,q) denotes the beta distribution with parameters p and q. It's p.d.f. is

$$f(x) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)}\, x^{p-1}(1-x)^{q-1} \qquad 0 < x < 1$$

$$= 0 \qquad \text{otherwise}$$

Be(1,1) is constant on the interval (0, 1/2) and is commonly called the rectangular, or uniform distribution on (0,1/2)

Cauchy(t) denotes the cauchy distribution with parameter t. The p.d.f. is:

$$f(x) = \frac{t}{t^2 + x^2} \qquad -\infty < x < \infty$$

where t is a positive scale parameter. The graph of this distribution resembles that of a Normal distribution, but the Cauchy distribution behaves more pathologically.

B. Relations Among Distributions

We will need to introduce some notation:

$\sim$	means "distributed as", or "has the distribution."
I.I.D.	is an abbreviation for "independently and identically distributed."
X	will be used to denote a random variable, and sometimes will be indexed as X_i.

$\{X_i\}_{i=1}^{n}$ denotes a finite sequence of random variables, or more informally, an ordered (finite) collection of random variables.

$\Leftrightarrow$ reads as "is equivalent to the following".

$F(c)=P[x<C]$ is the probability that the random variable x is less than C. F(c) is called the cumulative distribution function.

1. $X \sim N(\mu,\sigma^2) \Leftrightarrow (X-\mu)/\sigma \sim N(0,1)$.
2. If $X \sim N(0,1)$, then $X^2 \sim \chi^2(1)$.
3. If $\{X_i\}_{i=1}^{n}$ are IID $\chi^2(1)$, and $Y = \sum_{i=1}^{n} X_i$, then $Y \sim \chi^2(n)$.
4. If $X \sim N(0,1)$, $Y \sim \chi^2(n)$, and if X, Y are independent, and $Z = X/\sqrt{Y/n}$, then $Z \sim t(n)$.
5. If $X_1 \sim \chi^2(n_1)$, and $X_2 \sim \chi^2(n_2)$, and $Y = \frac{Y_1/n_1}{X_2/n_2}$, then $Y \sim F(n_1,n_2)$.
6. If $X \sim t(n)$, then $Y = X^2$ satisfies $Y \sim F(1,n)$.
7. If X_1, X_2 are IID N(0,1), then $Y = X_1/X_2$ satisfies $Y \sim$ Cauchy(1).
8. If X is t(1), then X is Cauchy(1).
9. If $X \sim u(-\frac{\Pi}{2}, \frac{\Pi}{2})$ and $Y = \tan X$, then Y is Cauchy(1).
10. $X \sim F(n_1,n_2) \Leftrightarrow \frac{1}{X} \sim F(n_2,n_1)$.
11. $X \sim \gamma(\lambda,n) \Leftrightarrow \lambda X \sim \gamma(1,n)$.
12. $X \sim \chi^2(n) \Leftrightarrow X \sim \gamma(1/2, n/2)$.
13. If $X \sim \gamma(\lambda, 1)$ then X has what is commonly called the "negative exponential distribution" with parameter λ (denoted exp λ).
14. If $\{X_i\}_{i=1}^{n}$ are IID exp λ, then $\sum_{i=1}^{n} X_i \sim \gamma(\lambda,n)$.
15. If $\frac{n}{m} Y \sim F(m,n)$, then $\frac{X}{1+X} \sim$ Beta(m/2, n/2).

APPLICATION OF ABOVE TO RND PROGRAM

With the modest selection of distributions provided and equipped with knowledge of the effect transformations have on a distribution (the above list provides a good start), many additional distributions may be obtained.

EXAMPLE 1. (Inverse Function method). Given an arbitrary continuous cumulative distribution function, $F(x)$ and its inverse, $F^{-1}(y)$ applied to a sequence $\{X_i\}_{i=1}^{n} \sim u(0,1)$ yields a sequence $\{X_i\}_{i=1}^{n} \sim F(x)$.

EXAMPLE 2. Knowing that $\gamma(\frac{1}{2}, n)$ is equivalent to $\chi^2(2n)$, and using no. 12 of the above list of relations, we may generate a first matrix of $\chi^2(n)$ random numbers, a second matrix of $N(0,1)$ random numbers, and by element wise dividing the second by the first obtain a matrix of $t(2n)$ random numbers.

APPENDIX E

Other Programs

In addition to the statistical programs in the SOUPAC system, the SOUPAC group maintains on OS/360 disk file a library of other programs which are briefly described below. Details about these programs and the means of accessing them are available at the SOUPAC Office.

PTSVIEW (Points of View Analysis)

This program performs a factor analytic points of view analysis following a procedure developed by Tucker and Messick (1963). At the specification of the user, the program computes either 1) cross-products; 2) covariances; or 3) correlations from the raw data (usually judgments). The analysis is then performed on this product matrix.

The program may be used for a second "pass" with both the original data matrix and a second "hypothetical subjects matrix" (containing coordinates of the idealized individuals) as input. The result is a matrix of hypothetical judgments, one set for each idealized individual; these are the judgments that, under the model, would have been made by each of the idealized individuals.

TØRSCA (Nonmetric Multidimensional Scaling)

This program performs nonmetric multidimensional scaling. The program computes a geometric representation of a data matrix such that the distances between the points in the representation best reproduce the order of the entries in the data matrix. The geometric representation may be in any Minkowski space (including city-block space and Euclidean space), and the order being reproduced may be the inverse of the order of the entries in the data matrix. Finally, the data matrix may be either a rectangular matrix or a symmetric matrix. In the former case, it is assumed that the space to be derived is a joint space of both row and column variables.

TPØLY (Least Squares Polynomial Fit)

Instead of fitting a polynomial in standard form to a set of data points by least squares, TPØLY fits a linear combination of Chebychev polynomials. The resulting normal equations are then solved by Cholesky's method. Both coefficients of the linear combination of Chebychev polynomials and the coefficients of the equivalent polynomial in standard form are calculated and printed.

This method avoids inverting a possibly ill-conditioned matrix and the round-off error properties are excellent. However, the variance-covariance estimates of the coefficients are lost.

UMAVAC (Univariate and Multivariate Analysis of Variance and Covariance)

UMAVAC performs univariate and multivariate linear estimation and tests of hypotheses for any crossed and/or nested design, with or without concommintant variables. The number of observations may be equal, proportional or disproportionate, the latter including missing observations and incomplete designs.

Among the possible analyses which can be performed by this program are regression analysis, including canonical correlation analysis and step-wise regression analysis, analysis of variance, and discriminant analysis.